Software Engineering Institute

Global Information Grid Survivability: Four Studies

Richard C. Ciampa
Dawn Day
Jennifer R. Franks
Christopher T. Tsuboi

March 2007

SPECIAL REPORT
CMU/SEI-2006-SR-008

CERT Program

Carnegie Mellon

This work is sponsored by the U.S. Department of Defense.

The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (http://www.sei.cmu.edu/publications/pubweb.html).

Contents

List of Figures

Acknowledgements

Our team would like to thank the following staff at the Software Engineering Institute for their help in advising us and providing valuable guidance toward the completion of this project: Bob Ellison, Tom Longstaff, Ed Morris, Pat Place, and Carol Woody.

In addition, the editing provided by Claire Dixon is greatly appreciated.

Executive Summary

The four studies in this document are student contributions to the SEI Global Information Grid (GIG) Survivability Study. Individual abstracts for the four student studies are below.

Autonomous Agents: The Good, the Bad, and the Underutilized

Richard C. Ciampa

Autonomous agents have been touted as the panacea to many information technology problems for over 30 years. Yet, despite their promise, they are not widely used. This study sheds some light onto the reasons why these much-discussed autonomous agents are so underutilized. The first section of this study is a review of autonomous agents and the conditions that facilitate their implementation. Since autonomous agents are not appropriate or viable in every situation, the challenge is to summarize the environmental requirements for successful use of autonomous agents. The second section explores why autonomous agents have succeeded or failed despite operating in an environment that is conducive to their use. The critical factors that should be used to measure success in agent use are examined. And finally, an attempt is made to extract some knowledge from the limited success achieved through the use of autonomous agents and to determine if autonomous agents could be utilized in the Joint Battle Management Command and Control (JBMC2) technology plan.

Component Testing and Its Limitation When Applied to a System of Systems

Dawn Day

Systems-of-systems (SoS) testing is an area in which no clear protocol or procedure has been developed. One method of verifying the functionality of the entire system is to test and certify individual components and then to test critical pathways to achieve some level of tested acceptability. This study identifies selected approaches to the testing and certification of components and how this process is limited in testing the overall complex system. First, the study describes how to select appropriate components for testing in a particular context, recognizing that each organization may define this term differently. Next, it presents issues to consider when selecting which components to test. Third, the study discusses risk-based testing and testing of COTS products. Finally, the study presents the limitations of component testing as it applies to the components within a system of systems.

Evaluating Emergence Issues Resulting from Unanticipated Consequences

Jennifer R. Franks

When completed, the Global Information Grid (GIG) will connect all information systems and their individual components and services within the Department of Defense (DoD) into one unified information system. In any effort to transition several individual systems into one cohesive structure, it is important to consider the unanticipated consequences that can occur. Unanticipated consequences are a series of unforeseen events that can be attributed to the architectural mismatches of systems with conflicting parts. Architectural mismatches can lead to several internal and external system complications, with the worst-case scenario complications ending in cascading failures. This study will show how architectural mismatches and cascading failures can occur. It will focus on some specific GIG system integration components and discuss where interactions are likely to result in emergent behaviors. This study will also outline some of the specific systems within the GIG that will be integrated, and provide supporting details to the types of unanticipated consequences that can occur when these systems are integrated. The GIG is not only demanding increased interoperability and integration of tools from several different systems, but a functionality and quality that is in accordance with a dependability level that will suit the DoD's needs as a whole.

IPv6 Network Addressing: Architectural Properties and Operational Survivability Effects

Christopher T. Tsuboi

One of the key Department of Defense mandates for next generation military data networks, including the GIG, is the employment of Internet protocol version 6 (IPv6). A crucial part of the IPv6 specification is its change to network addressing architecture. Addressing plays a critical role in both network architecture and operations and has a direct bearing on the survivability of network systems. This views system survivability as a composite of other system attributes and holds that the attribute most relevant to network addressing is operational control. Operational control is defined as a continuous cycle of three elements: cognizance, analysis, and execution capability. Operational survivability effects of IPv6 addressing result from the enhancement or degradation of these elements, in terms of several architectural properties, including: the increase in address space size; hierarchical routing and addressing; increased end-to-end transparency; increased transparency, autonomy, and complexity of end systems; and increased automation in the local network. The relationship between survivability, operational control and network addressing is discussed in Section 2. Section 3 contains a technological overview of IPv6 addressing architecture. Section 4 discusses five architectural properties of networks employing IPv6 addressing. These architectural properties are used as the framework for the survivability analysis in Section 4. Finally, Section 6 makes general observations about the operational survivability of IPv6-addressed networks and provides specific recommendations for IPv6 deployment in the GIG.

Abstract

The four studies in this document are student contributions to the SEI Global Information Grid (GIG) Survivability Study. Each study explores an issue relevant to the survivability of networks which are systems of systems. Since the GIG is inherently a system of systems, the survivability of operational concepts such as Joint Battle Management Command and Control (JBMC2) will largely depend on the extent to which GIG architecture is approached from this perspective. Systems of systems differ from large, monolithic systems because of the simultaneous independence and interdependence of their constituent parts, and therefore traditional survivability methods are not sufficient. To deal with the operational complexity resulting from qualities peculiar to systems of systems, planners and builders of the GIG will need to formulate broad strategic approaches taking these qualities into account. These four studies have attempted to identify characteristics of systems of systems which may be useful in this endeavor. The specific areas explored in this document include the following: the applicability of autonomous agents in a system of systems; the suitability of conventional software testing in a system-of-systems environment; emergent properties and unanticipated consequences in a system of systems; the role of ontologies in systems-of-systems interoperability; the architectural properties and operational survivability effects of internet protocol version 6 (IPv6) technology.

1 Autonomous Agents: The Good, the Bad, and the Underutilized

1.1 Introduction

This section neither engages in nor furthers the debate of artificial intelligence. It assumes that the agents discussed in this study have some degree of intelligence, whether real, artificial, or imagined, such that they would be capable of passing the Turing test [Turing 50].[1] The characteristics that typify autonomous agents are examined, along with the factors that mark their successful operation within a system-of-systems environment.

1.1.1 Characteristics

The idea of using autonomous agents is an appealing one. Tell them what you need, set them loose, and reap the benefits when they achieve their goals. But what are autonomous agents?

Brustoloni characterizes autonomous agents broadly as “systems capable of autonomous, purposeful action in the real world” [Brustoloni 91].

- *purposefulness*: agents need to have a goal toward which they aspire, an intention to find a thread of existing services that can be used to satisfy a particular desire or a plan of action which to implement

Autonomous agents also need a foundation of information with which to operate. As a carpenter fills his or her tool box with tools necessary for a particular job, agents must have their tool boxes packed with things they need to know in order to learn what they need to discover. For example, it would be difficult to try to learn about the multiplication function without an understanding of how the addition function works. What does the agent need to know to achieve its goals? Dastani states that “the reasoning capability of agents is implemented by means of various rule bases. In particular...goal rule base, an intention rule base, and a plan rule base” [Dastani 03].

- *rule base*: the collection of knowledge which agents utilize to define their goals and plan their courses of action

[1] The Turing Test evaluates a machine’s capability for performing human-like conversation. It was proposed by Alan Turing, British mathematician, logician, and cryptographer, who is considered the father of modern computer science. In his test, a human judge engages in a natural language conversation with two other parties, one a human and the other a machine; if the judge cannot reliably tell which is which, then the machine is said to pass the test.

However, autonomous agents should not merely be hard coded to follow a predetermined course of action. In order to operate and successfully achieve their goals, agents, above all, must be flexible. Wooldridge and Jennings (W & J) identify two characteristics of flexibility, social ability and reactivity, as the following [Wooldridge 95]:

- *reactivity*: agents perceive their environment, (which may be the physical world, a user via a some interface, a collection of other agents, the Internet, or perhaps all of these combined), and respond in a timely fashion to changes that occur in it
- *social ability*: agents interact with other agents (and possibly humans) via some kind of agent-communication language

OWL-S, the successor to DAML-S, supplies autonomous agents with an ontology which facilitates the functionality of social ability. OWL-S provides for the construction of profiles for alerting other agents of their capabilities and finding other agents which possess needed services. The controlled vocabulary of OWL-S also provides agents with detailed descriptions of agent operation and how the different agents should communicate with one another.

In addition to flexibility, an autonomous agent must be just that, autonomous. This study uses a consensus definition of autonomy found in Rudowsky as the following [Rudowsky 05]:

- *autonomy*: the ability to act without in intervention of humans or other systems, and have some kind of control over their actions and internal state

A simple example of an agent would be a home’s heating system. The furnace collects temperature data from a thermostat elsewhere in the house and either turns the heat on or off depending upon the current room temperature. This may not be a very intelligent agent but it is certainly acting autonomously. This simple agent also demonstrates a social ability by communicating with the thermostat and reactivity because it exhibits a perception of its environment. It reacts to the changes within its surroundings.

What truly makes an autonomous agent so powerful and desirable is the sixth characteristic, pro-activeness, which Wooldridge and Jennings define as follows [Wooldridge 95]:

- *pro-activeness*: agents do not simply act in response to their environment, they are able to exhibit goal-directed behavior by taking the initiative

A pro-active agent would not only collect information about its environment but comprehend and reason about what that information means. To revisit the heating system example, suppose a door has been left open and the temperature within the house never rises high enough to shut off the furnace. Where a non pro-active agent may let the furnace run continuously, a pro-active agent may recognize that there is a problem, since a certain amount of time has passed by which the house should have been adequately heated. The proactive agent might either notify the homeowner or shut down the furnace as a preventative measure.

1.1.2 Environment

Russell and Norvig classify agent environments based upon different properties that have an effect on the agent's decision making process [Russell 95]. They are as follows:

- Deterministic vs. Non-deterministic

  Most reasonably, complex systems are non-deterministic—the state that will result from an action is not guaranteed even when the system is in a similar state before the action is applied. This uncertainty presents a greater challenge to the agent designer.
- Accessible vs. Inaccessible

  An accessible environment is one in which the agent can obtain complete, timely and accurate information about the state of the environment. The more inaccessible an environment, the more complicated the environment, the more difficult it is for agents to operate within it.
- Static vs. Dynamic

  Static environments remain unchanged except for the results produced by the actions of the agent. A dynamic environment has other processes operating on it thereby hanging the environment outside the control of the agent. A dynamic environment obviously requires a more complex agent design.
- Discrete vs. Continuous

  If there are a fixed and finite number of actions and percepts—impressions obtained by use of the senses – then the environment is discrete. A chess game is a discrete environment while driving a taxi is an example of a continuous one.

The Joint Battle Management Command and Control (JBMC2) will operate certainly within a very unforgiving environment.

Even if we can assume a deterministic environment where similar actions will return similar results, the environment will be a highly inaccessible one. Much can be done to collect accurate information but it can be assumed that some data collected will be inaccurate, incomplete, unavailable or all three.

The JMBC2 technology plan is to include mission threads requiring "collaborative system activities across strategic, operational, and tactical organizational levels linking Services, Agencies, and Combatant Commands." Each mission step "consists of a 'contract'...with the preceding and following steps...[which]...is not necessarily static but may have to be negotiated at run-time to reflect the situational conditions."[2] This obviously describes a dynamic environment since other processes will affect the environment in which the agents operate.

Depending upon the scope of JBMC2 functions, the environment in which its autonomous agents will operate will be somewhere on the continuum between discrete and continuous. In May 1999, the National Aeronautics and Space Administration (NASA) successfully

[2] Software Engineering Institute. "OSD AT&L JFI GIG Survivability Study." CMU/SEI Presentation. December 5, 2005. Not publicly available.

executed the Remote Agent Experiment, demonstrating the applicability of autonomous agent technologies to spacecraft command and control [Bernard 99]. Although there were thousands of combinations of actions and sensor readings used to navigate and control the spacecraft, this number of combinations was fixed. As in the chess game mentioned in the term’s definition, there may be a large number of possibilities, but it is a calculable number. If JBMC2 agents are designed to perform and to sense too much, this may create an environment which is continuous and ultimately more complex.

Each environmental property adds a level of difficulty in the design and operation of autonomous agents. Together, they heighten the amount of complexity by geometric proportions.

1.1.3 Complexity

Complexity inherently arises from the dynamics between both the characteristics of autonomous agents and the environment in which they operate. As more is expected of the autonomous agent, its rule base increases, and more uncertainty is introduced in its environment, the complexity of the agent’s implementation grows. Perme and associates speculate that the “key success factor in the use of intelligent agents is to define them such that the complexity of the rule base does not outweigh the value gained by their use. Seeking a complex rule base to address all issues would have resulted in failure” [Perme 03]. As Figure 1 illustrates, if the characteristics and environmental elements of autonomous agents are viewed as building blocks, as the weight of the blocks moves away from stable, known circumstances, the stack becomes unstable. If an agent is created to run in an inaccessible, non-deterministic, dynamic, and continuous climate the time and effort required to design, run, and maintain an autonomous agent may become unwieldy and be susceptible to unexpected consequences.

Communication between autonomous agents adds another layer of complexity. In addition to alerting other agents of their capabilities and finding other agents which possess needed services, agents must be able to exchange data. To do this, agents must overcome the differences in data sets, hardware platforms, operating systems, and so forth. They must navigate between numerous agent communication languages (ACL) without an understanding of the ontological knowledge of the other agents. “Furthermore...two ACLs cannot really be used in an open environment (i.e., the Internet) to communicate with a shared understanding of the meaning of the terms in messages” [Qiu 05]. Since there is no universal standard ontology, complexity of some form exists that increases exponentially with every agent added to the process.

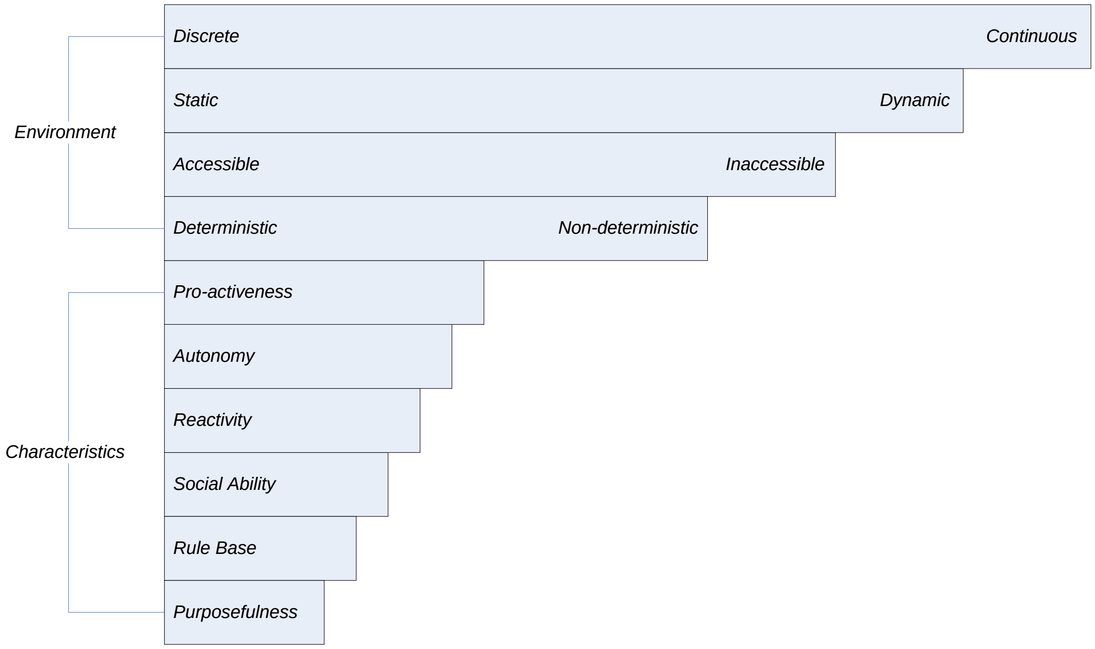


Figure 1: Complexity Building Blocks
As more is expected from the autonomous agents, their complexity topples the stack.

This is not to say that complexity will guarantee failure. With NASA's Remote Agent Experiment, a primary objective of the experiment was to demonstrate the viability of onboard spacecraft autonomy. NASA ran a series of experiments specifically to "demonstrate numerous autonomy concepts ranging from high-level goal oriented commanding of on-board planning to robust plan execution to model-based fault protection" [Bernard 99].

1.2 Critical Success Factors

In some cases where an organization implements some form of autonomous agent within a larger project, the organization sets expectations to be specific to the outcome of the individual application without regard for how the autonomous agents affect the organization as a whole. An agent that produces desired results locally may create unanticipated and undesired results in other areas of the system.

1.2.1 Current Critical Success Factors

It appears that current success is not measured in terms of how well autonomous agents function. The agent is viewed as a tool, such as a hammer, but with a certain level of abstraction. However, if the goal is to open a bottle of wine, a hammer will satisfy the critical success factor quite nicely but might leave the wine undrinkable. In this analogy, either the hammer should have been evaluated for appropriateness or the critical success factor should have been more specific.

1.2.2 Increased Processing Speed

The impact that use of autonomous agents has on processing speed has been subject to controlled study. The military ran a series of experiments with the Domain Adaptive Information System (DAIS), a mobile autonomous agent system for information discovery and dissemination in a military intelligence network. Hoffman, states that the "operational goals of the military intelligence (MI) operation...are to ensure that critical information is collected, represented correctly, and rapidly disseminated to the appropriate decision makers" [Hoffman 98]. The report concluded that DAIS greatly increased the delivery rate and decreased the delivery delay of the distributed information for military intelligence.

1.2.3 Managed Complexity

The autonomous agent's capacity to perform complex tasks has also been examined. In another example, the Intelligent Distribution Agent (IDA), a "conscious" software agent that was developed for the U.S. Navy, is tasked with assigning a sailor to a new tour of duty at the end of the old tour [Franklin 03]. Success was measured, in this case, by a successful reassignment and how closely the IDA mimicked the actions of a human detailer charged with the task of reassigning naval personnel. This task involved synchronizing the touring activities of 280 individuals.

1.2.4 Shortcomings

In both of the above examples, several factors were not considered when measuring the success of autonomous agents. For example, the costs associated with using autonomous agents were not considered. Hoffman and associates reported that the information delivery rate increased from below 50% to 100% [Hoffman 98]. But at what cost? Spending $100 million to save $1 million a year would be foolhardy since it would take 100 years to make it worthwhile. In addition, there was no metric to measure the accuracy of the information being processed. And, although user training is considered in the report findings, there is no mention of whether or not the users had accepted the agents' output. If they did not, the speedier deliver would be meaningless.

A more effective set of critical success factors should be considered for determining whether or not autonomous agents make sense.

1.3 Better Critical Success Factors

The success of autonomous agents can be better gauged if the agents themselves are evaluated. Rather than assume the agents are successful because the system as a whole is successful, one should assess each agent's success individually. This can be accomplished through the use of traditional metrics such as security, return on investment, data integrity, and user acceptance. These metrics are discussed below.

1.3.1 Security

Rudowsky asks “will data be secure, particularly in a distributed environment? Will an agent respect restrictions from other servers and go only where allowed?” [Rudowsky 05]. It appears that security is often an afterthought. Hoffman’s discussion of battlefield mobile agents includes the statement that the collection of a “quantitative operational measure was hampered by the sensitivity of the data...” [Hoffman 98]. However, no mention is made of testing whether or not security is breached or weakened. One must assume that in a military operation security would be paramount. Autonomous agents must be expected to operate effectively with the highest of security measures in place.

1.3.2 Return on Investment

Measuring the value of an investment is a big part of business planning. A return on investment calculation is a small metric expressed as a dollar value or percentage of dollars invested. While a return on investment calculation for a particular investment might be negative, the intangible benefits could justify the expense. Return on investment is an important metric, but it needs to be balanced with a rigorous analysis of all factors.

In the IDA example, Franklin admits that return on investment may not have been a priority for the project when he states that “currently running IDA requires the most powerful processor generally available, and an almost full complement of RAM” [Franklin 03]. In addition, no mention is made about the personnel required to administer the IDA system.

If an analyst is required to expend as much time setting up the system, monitoring its actions, and making corrections as would have been spent constructing and maintaining the database manually, we are no better off [Carrol 02].

1.3.3 Data Integrity

Maes points out that autonomous agents cannot tell the difference between correct and incorrect data by themselves [Maes 94]. Creating an agent with the ability to understand incorrect or bad information would be an extensive exercise in harnessing artificial intelligence. In the DAIS project, the delivery rate, delivery delay, acknowledgements and correct formatting were used to demonstrate the success of the engaged autonomous agent. Hoffman and associates neglected to monitor whether or not the information was correct [Hoffman 98].

Barber and Park contend that the certainty of the information influences an autonomous agent’s decision making process [Barber 03]. In addition, the reputation of the data source—the level of confidence an agent has in the quality and the reliability of information—plays a big part in an autonomous agent’s ability to function as planned.

1.3.4 User Acceptance

Hoffman contends that one needs “...to gain user acceptance to measure system success” and this may be one of the biggest obstacles to overcome [Hoffman 98]. For one to give up control of one’s home thermostat is relatively easy. However, to give up control of where and when one’s artillery will be fired might be relatively difficult.

Popular culture is full of tales of autonomous agents created to help humanity which ultimately end up enslaving or killing humans. In the film *2001: A Space Odyssey*, a monumental space expedition is mounted with two astronauts operating a huge spaceship with the assistance of the latest and most complex computer, HAL [Kubrick 68]. HAL is a revolutionary computer system with artificial intelligence that is equal or perhaps superior to a human’s. For the sake of the mission, HAL risks the lives of the crew to protect itself. Autonomous agents have also been frequently featured in the work of author Isaac Asimov, where they are created to protect mankind but ultimately wind up controlling it.

These fictional examples are believed far fetched by many but the fear of agents’ exhibiting too much autonomy is real. Suppose, with the earlier thermostat example, the system is set up to not only regulate the heat but to operate the entire home. Smart houses analyze various activities detected by the sensors located throughout the house. Information can then be used to assist the homeowners directly or passed on to others. Would home owners want their houses to know that much about them? Suppose it could be determined that heating costs go down when the home owner receives a delivery from Amazon.com. The autonomous agent would already have access to the home owner’s credit card, used to pay bills automatically, and could purchase a nice little present to minimize the cost of heating the house.

Most stories of computers running amok are apocryphal but the hesitation to relinquish control is real. Kephart and Chess assert that “...autonomic systems will be extremely valuable, and their value will increase substantially as autonomic computing technology improves and earns greater trust and acceptance” [Kephart 03].   In addition, Bradshaw states that “an impediment to the widespread adoption of agent technology is social—for individuals to be comfortable with delegating tasks they must first have trust in agents” [Bradshaw 97].

1.4 Good for the JBMC2?

In the Defense Advanced Research Projects Agency (DARPA) report *Considerations in Developing Survivable Architectures for Global Information Grid (GIG) Systems,* two proposed systems incorporate the use of autonomous agents [DARPA 01]:

1. Active Trust Management for Autonomous Adaptive Survivable Systems, which builds self- monitoring and adaptive systems that detect failures, infer underlying compromises, and steer computations away from compromised or questionable resources.

2. Fault-Tolerant Networking Through Intrusion Identification and Secure Compartments (FINIISC), which demonstrates how to partition the network into autonomous components to allow any component to fail without affecting the entire network.

Will users be confident that an untrustworthy agent will disclose its untrustworthiness? Both applications, though security based, may lack the trust necessary for user acceptance. It may be difficult for people to give control of security operations to an autonomous agent. If an agent as been compromised then it merely needs to advise other agents that it hasn't been compromised to continue its misdeeds.

Depending upon the environment in which the agent will be working, the added complexity of design may render the autonomous agents unworkable or not worth the investment. Especially in the area of security, additional complexity may make the solution a greater threat than the problem one is trying to solve. Data sensitivity can hinder the ability of autonomous agents and as Hoffman and associates point out, the "Collections of quantitative...measures was hampered by the sensitivity of the data" [Hoffman 98].

1.5 Conclusions

Autonomous agents are not the silver bullet with which most problems can be resolved. Although the technology relevant to creating autonomous agents is over thirty years old, it is still a relatively new and untested one. "Developers should be aware of the downside inherent in the use of new and untested technology" [Rudowsky 05].

Autonomous agents should be used for specific tasks with great constraint. In a narrower environment, one can somewhat reduce the risks and uncertainties associated with environmental elements. Autonomous agents are often touted as mechanisms to reduce complexity but if the breadth of the agents' operational environment is too great, they may end up creating a greater degree of complexity in their application. Agents are used successfully on eBay, the online auction web site, because their goals are simple and their scope is limited. The eBay agents act under the same principles as reactive thermostats do and have strict constraints regarding how much to bid at given intervals.

Success should not be measured with one or two metrics but to the extent that will best illustrate the overall appropriateness of using autonomous agents in a situation. If autonomous agents are to be widely used within the JBMC2 technology plan, they must be challenged to function effectively and under the worst conditions.

2 Component Testing and Its Limitations When Applied to a System of Systems

2.1 Introduction

Consider the system described below:

> The Global Information Grid (GIG) is defined as the globally interconnected, end-to-end set of information capabilities, associated processes, and personnel for collecting, processing, storing, disseminating, and managing information on demand to warfighters, policymakers, and support personnel.[3]

The complex system that will result from its development can be termed a system of systems (SoS). In this study, SoS refers to conjoined systems that have operational and managerial independence. Its design, testing, and implementation present a unique set of challenges. This study seeks to address problems and issues that arise from component testing in an SoS environment. The complex nature of an SoS places definite limitations on the component testing approach. Although testing has commonly been viewed as a post-development task, this study will argue that post priori testing is not suitable in an SoS environment. In SoS, testing must also take place before and during development in order to formulate useful test cases. The study also describes conventional component testing and certification processes in detail and identifies where these processes may fail in an SoS environment. The set of criteria used to certify and evaluate any system acquired by or designed in the federal government is outlined in the DoD Information Technology Security Certification and Accreditation Process [DoD 97]. This study assumes this same set of criteria will be used for certifying the components of the GIG and is therefore worth discussing here.

In particular, based on DoD stated goals, this study includes consideration of the use of commercial off-the-shelf (COTS) products in an SoS and ways for identifying and testing other components of the SoS, such as through use of a risk-based approach. Specialized testing methods for SoS have not yet been clearly defined, and so it might seem natural to use existing methods. This study argues that existing methods have limitations that should be understood and carefully considered before being used in an SoS context. It is important to first define an SoS in the context of this study before addressing the issue of testing in its environment.

[3] http://en.wikipedia.org/wiki/Global_Information_Grid.

2.2 Systems of Systems (SoS)

It is important to define how an SoS is different from other systems in order to address why current testing methods should not be applied “as-is” to SoS. Therefore, this study focuses on the characteristics of systems of systems as identified by Mark Meier and how they are distinguished from complex but monolithic systems [Maier 96]. He explains that systems of systems are distinguishable from large, monolithic systems by the following:

1. the operational independence of their components
2. the managerial independence of their components
3. their evolutionary nature
4. their emergent behaviors
5. a geographic extent that limits the interaction of their components to information exchange

This study will focus only on three of Maier’s five defined characteristics as important to testing considerations: operational independence, managerial independence, and evolutionary development. Operational independence of the elements indicates that “if the system of systems is disassembled into its component systems the component systems must be able to usefully operate independently.” Managerial independence of the elements signifies that “the component systems not only can operate independently, they do operate independently. The component systems are separately acquired and integrated but maintain a continuing operational existence independent of the system of systems.” Finally, evolutionary development exists because the “system of systems does not appear fully formed. Its development and existence is evolutionary with functions and purposes added, removed, and modified with experience” [Maier 96]. These characteristics indicate that testing for an SoS, because it will be different from a closed monolithic system, has not been well defined.

One approach to testing an SoS is to use component testing. Component testing focuses on the individual components of a system and tests them to determine a fitness of use in the overall system. In a monolithic system, component testing is relatively effective since components are defined through a decomposition of the system as a whole. It relies on the idea that if each of the parts of a whole is certified then the whole is certifiable. It is this assumption that reduces the effectiveness of this testing method when applied to an SoS. In an SoS, the components themselves may be systems composed of a number of sub-components that also need to be tested. While component testing can be used in an SoS environment, it has limitations due to its complex nature and the number of pathways created by the interconnection of its components.

The GIG is envisioned as a very large SoS. It will comprise existing information technology and national security systems, which currently are maintained by different agencies across the federal government and military. Due to the GIG’s complex nature and interconnection of components, testing must be done as effectively as possible in order to evaluate risks. The remainder of this study focuses on describing the component testing and certification process and the limitations of these methods when applied to an SoS.

2.3 What is a Component?

As previously stated, in an SoS the components may themselves be systems with sub-components which may also be rather large in size. In order to begin any component testing, the first step is to establish the characteristics that will identify the components in the overall system. These characteristics can be different for any given system, based on the testing objectives. While there are no clear checklists that identify a component specifically for any application, there is one characteristic that must apply to any selected component: it must be closed. The term “closed” in this context refers to the fact that the component can be isolated and will be able to carry out some function or produce an output.

In the context of an SoS, it can be difficult to define a component; an individual system can be a component and for each system within the system of systems there can be many components. A component may be

> a class or a cluster of classes that are tightly coupled, but it is always a conceptually atomic unit. It is the component testing perspective that is important and not the size of the pieces being tested. [This] perspective views the [component] being tested as intended for integration with other pieces rather than as a complete system in itself. This both helps to determine what features are tested and how they are tested [McGregor 97].

Component testing requires the appropriate and exact defining of the component in the context of the particular environment that will contain it.

2.3.1 Which components to test

Once this definition has been established, development of test cases for each component can begin. Based on the size of the SoS, it may be inconceivable to test all cases for every component. It is useful to establish a guideline for which test cases are prioritized and can be selected given inadequate time or resources to test each case. Each component must be tested, but not every component must be tested to the same level of certification.

Because SoS components function independently as well as interoperate with other components, they cannot all be tested to the same extent. It is necessary to devise some standard way of prioritizing which systems are tested and to which degree. McGregor’s recommendation is to “select a component for testing when the penalty for the component not working is greater than the effort required to test it” [McGregor 97]. In a system as large as the GIG, however, criteria might require much more specific definition. Moreover, “There are several situations in which the individual components should be tested regardless of their size or complexity…” as in the following cases:

- reusable components. Components intended for reuse should be tested over a wider range of values than a component intended for a single focused use.
- domain components. Components that represent significant domain concepts should be tested both for correctness and for the faithfulness of the representation.
- commercial components. Commercial off-the-shelf components should be internally tested for acceptance for use within the given system [McGregor 97].

Guidelines that will help testers determine which components to test and how to test them should be developed, ideally, before development begins. Testing should be an intermittent process as development is taking place and components are forming that will be integrated into the SoS. As testers cannot test all cases for each component, the guidelines must also specify the level at which a test is passed or failed. As no component will be completely defect-free, the notion of certification is generally used, as explained below.

2.4 Component Certification

"Component certification is the process of evaluating software components according to a predefined set of criteria. The set of criteria that is used in component certification allows the user of the component to judge its quality beforehand."[4] The certification process can also be applied to any component within an SoS, given an appropriate set of criteria. Once the components have been identified, the necessary criteria can be used to evaluate them. Standardized certification criteria are set forth by an external authorized organization.

The dictionary definition below gives us insight into the conventional meaning of certification and exposes what may be a source of many hidden assumptions about how it does, or should, relate to components: *1. to attest as certain; give reliable information of; confirm…2. to testify to or vouch for in writing…3. to guarantee; endorse reliably; to certify a document with an official seal* [Webster 96].

Below is a closer examination of these definitions.

- *To attest as certain* suggests an authoritative testimonial or "to stand by" a statement.
- (To) *give reliable information* suggests an objective quality.
- (To) *confirm* suggests a corroboration of other information.
- (To) *vouch for in writing* suggests evidence and legal implications.
- *To guarantee* suggests contractual obligations and remedies.
- (To use) *an official seal* suggests a designated authority [Wallnau 04].

Component certification is useful in an SoS environment to ensure a certain level of quality and risk-aversion. As it would be impossible to have a risk-free component, certification guarantees that the component is suitable to a specific set of criteria chosen with the complex nature of the SoS in mind. Certification also allows for some flexibility; the set of criteria to which the component is compared can change and evolve for each certification process.

[4] http://se.ethz.ch/people/bay/publications/2004/research_plan_till_g_bay.pdf.

2.4.1 DoD Information Technology Security Certification and Accreditation Process (DITSCAP)

The DoD Information Technology Security Certification and Accreditation Process (DITSCAP) establishes a standard DoD-wide process, set of activities, general tasks, and a management structure to certify and accredit information systems (IS) that will maintain the information assurance (IA) and security posture of the Defense Information Infrastructure (DII) throughout the life cycle of the system. Certification involves comprehensive evaluation of the technical and non-technical security features of an IT system and other safeguards, conducted in support of the accreditation process, to establish the extent that a particular design and implementation meets a set of specified security requirements. It involves security analyses in the areas of physical, personnel, administrative, information, information systems, and communications. Accreditation involves a formal declaration by the Designated Approving Authority that an IT system is approved to operate in a particular security mode using a prescribed set of safeguards at an acceptable level of risk. This process consists of four phases:

1. the definition phase, in which key players agree on the intended system mission, security requirements, Certification and Accreditation (C&A) boundary to establish the closed component for evaluation, schedule, level of effort, and required resources
2. the verification phase, which involves verifying the system's compliance with System Security Authorization Agreement (SSAA) requirements. The goal of this phase is to confirm an integrated system for certification testing and accreditation.
3. the validation phase, which requires that the system be on hand. The goal of undergoing this phase is to obtain full approval to operate the system (accreditation) by validating the system's compliance with SSAA requirements.
4. the post accreditation phase. Its purpose is to assure maintenance of an acceptable level of residual risk.

Of particular importance to the scope of this study is the validation phase. As stated above, it is the results of this phase that indicate whether or not the system is approved for use. This phase involves rigorous testing and evaluation methods that help to provide a complete picture of the output of the system given certain inputs as well as any risks associated with the use of the system.

This phase incorporates sub-tasks such as security test and evaluation (ST&E), penetration testing, contingency plan evaluation, and risk management review.

The ST&E involves assessing the implementation of security requirements. The specific tasks include the following:

- Assess implementation of security design.
- Ascertain that CIAA (Confidentiality, Integrity, Availability and Accountability) are implemented as documented in SSAA and perform properly.
- Validate correct implementation of identification and authentication (I&A), audits, access controls, object reuse, trusted recovery, and network connection rule compliance.

- Evaluate system conformance with requirements, mission, and architecture as defined in SSAA.

Penetration testing involves testing for vulnerabilities for exploitation that could represent insider and outsider threats. The contingency plan evaluation examines planning for backup and continuation of operations (COOP). Finally, the risk management review checks for acceptable risks to CIAA.

Other tasks of the validation phase include: Communications Security (COMSEC) compliance evaluation, system management analysis, and a site accreditation survey. At the end of the phase, a certificate report is developed and a recommendation for accreditation is given or withheld.

In addition to employing the testing techniques described above, it might be useful to identify other methods for testing the components when these methods are not available or suitable, such as for the testing of COTS components for which no source code is available.

2.4.2 How to test the components

Components can be tested in two ways.

1. An artificial system environment can be constructed for the purposes of the test (known as a test harness); such a test is artificial.
2. The component can be tested within one or more of the systems for which the component was designed. Such a system must be closed for the purposes of the test, making the test artificial. Furthermore, the component is only tested on a subset of the possible systems for which it might be used in the future, making the test partial. Risk-based testing and COTS component-based testing are useful approaches for this purpose.

2.4.2.1 Risk-based testing

Risk analysis is the prediction of a possible outcome based on the probability of that outcome resulting. Risk-based testing involves identifying or prioritizing the test objects based on the damage that could occur from a failure or malfunction and the probability that failure or malfunction will occur. For the purposes of this study, the test objects are the components themselves. Risk can be assessed using any suitable method, such as assigning a low, medium, or high value to each component. The remainder of this study assumes this method of assigning risk levels. Once the risk levels are assigned, the appropriate extent of testing can be applied to each level. For example, for components with a high risk level, thorough testing (as defined by the actual testers) can be done. For medium-risk components, ordinary testing can be done, and so on.

Risk-based testing allows an organization to prioritize the components to be tested as well as the test cases to use on those components. Risk analysis can be applied to the task of identifying which parts of the overall system to test more intensely than the rest. The risks that are considered can be either technical or business related. In order to accurately assess these risks it is necessary to analyze the system's requirements and other requirements (such as those associated with the DITSCAP process) related to the development of the system. This analysis results in a mapping of the risks identified at the requirements level to individual uses cases. Each use case is assigned a risk classification. All of the use cases within a category are tested to the same level of coverage. The coverage levels must also be defined. The risk classification of the use cases can be mapped onto the components. Thus not all components will be tested to the same coverage level.

One criterion that could be used in a component-level risk analysis is whether the component is intended for reuse. The level of risk for the component increases when that component is integrated into a complex system. The risk is not only associated with the function of the component itself but also with its impact on the components with which it interacts. This increased risk comes from the expectation that the component must respond correctly and to a much wider range of inputs. Other criteria that could increase risk include the language features required for implementation of the component (using a newer language feature is a higher risk), complexity of the specification, and the maturity of the development environment including the tools and the personnel.

Once the use cases have been classified according to risk, the domain components referenced in each use case can be assigned the same risk classification as that of the use case. Assigning a risk value to the component requires a mapping strategy. One common conservative strategy is the maximum value strategy. In the case of security, the maximum risk associated with any of its use cases is a value of "high," so the component is given a value of "high."

This approach is effective, as the risks associated with the failure of critical components are directly related to critical missions of the DoD. This approach to testing allows for the consideration of these risks before implementation of the system. If considered beforehand, the use cases and associated risks determined can assist in the development process to produce a more secure and less risky system.

> Standard approaches to testing are implicitly designed to address risks. You may manage those risks just fine by organizing the tests around functions, requirements, structural components, or even a set of pre-defined tests that never change. This is especially true if the risks you face are already well-understood or the total risk of failure is not too high [Michael 05].

The GIG's intended use demonstrates that the total risk of failure is very high. In fact, the risk of small errors could impact the lives of many. Risk-based testing for a critical system such as the GIG would seem superior to other testing methods as it explicitly considers the risks and capitalizes on its ability to prioritize the risks as they relate to components in order to facilitate testing.

2.4.2.2 Commercial Off-the-Shelf (COTS) Product Testing

Commercial off-the-shelf (COTS) refers to “things that one can buy, ready-made, from some manufacturer’s virtual store shelf (e.g., through a catalogue or from a price list)” [SEI 97].[5] “The GIG will use commercial technologies augmented to meet DoD’s mission-critical user requirements” [NSA CSS 06].[6] Due to the unique nature of the GIG’s mission and specifications, significant testing will be needed for any acquired COTS product. COTS product testing, as it exists within the scope of component testing, consists of component qualification and component adaptation.

Component qualification is a process of determining “fitness for use” of previously-developed components that are being applied in a new system context. Qualification of a component can also extend to include qualification of the development process used to create and maintain it (e.g., ensuring algorithms have been validated, and that rigorous code inspections have taken place). This is particularly important in the context of applying COTS products to the GIG, as the majority, if not all, of the testing is done by the vendor. Qualifying the process can partially compensate or allow for confidence in upgrading that particular product in the future. Qualification involves both discovery as well as evaluation.

In the discovery phase of component qualification, the properties of a component are identified. Such properties include component functionality (what services are provided) and other aspects of a component’s interface, such as the use of standards. These properties also include quality aspects that are more difficult to isolate, such as component reliability, predictability, and usability.

Evaluation typically involves a combination of study-based studies of the components, discussion with other users of those components, and hands-on benchmarking and prototyping. The hands-on approach is unlikely in the case of COTS products, and this reduces the effectiveness of evaluation. There are techniques available to assist in formulating the evaluation of such products. The International Standards Organization (ISO) describes general criteria for product evaluation. There are also techniques that take into account the needs of particular application domains, such as those described by the Institute of Electrical and Electronics Engineers [IEEE 98].

Component adaptation can be necessary because individual COTS components are written to meet different requirements, and are based on differing assumptions about their context. The components must therefore be adapted when used in a new system based on rules that ensure that conflicts among components are minimized. These rules must be established in the planning phase of the GIG’s interconnection of components. If these rules are set before acquisition occurs, it will reduce the chance of failure or waste of resources.

[5] http://www.sei.cmu.edu/str/descriptions/cots_body.html.

[6] http://www.nsa.gov/ia/industry/gig.cfm?MenuID=10.3.2.2.

The degree to which a component's internal structure is visible suggests the need for different approaches to adaptation:

- white box, where access to source code allows a component to be significantly rewritten to operate with other components
- grey box, where source code of a component is not modified but the component provides its own extension language or application programming interface (API)
- black box, where only a binary executable form of the component is available and there is no extension language or API [Valetto 95, SEI 05][7]

Given the direction of the Department of Defense's acquisition procedures, it is reasonable to assume that the GIG vision will employ a large number of COTS components in order to achieve its functionality. The use of these components saves time and money initially; however they can impose stringent limitations on the overall functionality and security of the SoS. Further, each can impede testing, since for most COTS products the source code is not released and the product is not tailored specifically to meet the standards of the system it will become a part of. Component testing, in general, also breaks down in an SoS environment as the individual testing of components does not consider the interoperability and multiple pathways created by their interconnection.

2.5 Limitations of Component Testing in SoS

Component testing is useful when certifying a closed, functional unit. Using testing methods such as risk-based testing and testing of COTS products, one can identify whether or not the component has reached a suitable level of defect-free functionality. An SoS is composed by interconnecting many components. It is due to this interconnection that component testing breaks down in an SoS environment.

As a closed unit, the component has a finite number of inputs and corresponding outputs. When connected to another component, the inputs may differ from those in the closed state; this is particularly true when the component providing the input through its own output produces an unexpected result. Further, once components are integrated into a complex SoS, it is no longer sufficient to analyze the inputs and outputs. The analysis should now focus on the pathways created and the interoperability of components. Interoperability is "the ability of two or more systems or components to exchange information and to use the information that has been exchanged."[8] The results of component testing alone cannot give necessary insight into the behavior of the overall SoS.

Further, there are limitations of component testing when applied to COTS products. The primary limitation is that these systems are not open, meaning the source code is unavailable for viewing. This makes testing difficult as it reduces the product to a black box. Another

[7] http://www.sei.cmu.edu/str/descriptions/cbsd_body.html

[8] http://www.sei.cmu.edu/str/indexes/glossary/interoperability.html

limitation is the fact that testing is done primarily by the vendor. This means that testing is done with respect to the vendor’s intended use which does not necessarily include the environments or factors needed to ensure that it is suitable for the system into which it will be integrated. The effects of these limitations are further dramatized when COTS components are connected with other components. As the majority of COTS products do not include source code, it is difficult to design an appropriate interface with which the two components can communicate.

2.6 Conclusion

Currently, testing is considered and test cases developed only after the product has been developed. In a complex system of systems such as the GIG, this approach to testing is not suitable as testing must be considered in the development of such a project. An SoS is a complex environment in which to develop and test. The large scale of the components and functionality makes it difficult to identify appropriate test methods. Component testing can be applied to the SoS as part of the overall testing procedure. Component testing results in acceptance or modification of the components in the context in which they will work. The components which are COTS are currently tested primarily by the vendor based on specifications which do not necessarily match those of the intended environment.

This author recommends that testing focus on the components using a risk-based approach for determining the degree to which the components should be tested. However, component testing is not sufficient when used as the sole testing method. In an SoS environment, it is necessary to use more than one test method, especially when working with a critical infrastructure such as the GIG. Component testing may be coupled with interoperability testing to provide greater coverage of the risk factors and provide a broader view of the risks and functionality of the SoS.

3 Evaluating Emergent Issues from Unanticipated Consequences

3.1 Introduction

When completed, the Global Information Grid (GIG) will connect all information systems and the individual components and services within the Department of Defense (DoD) into one unified information system. The GIG system, with its communication and information assurance capabilities, will be a net-centric system operating in a global context that will provide authorized users with a secure and interconnected information environment. It is projected to meet near real-time needs of both the warfighter and the business user, for the purposes of providing the information processing, storage, management, and transportation needs to support all of DoD information systems [NSA 05].

The systems of the DoD that will be integrated to compose the GIG will include all owned and leased communications, computing systems and services, software, data, and security services, as they become unified to maximize information superiority. Providing support, operation locations and functioning capabilities, the GIG will support all of DoD, national security, and other related intelligence community missions and functions [Miller 01]. The GIG system integration will take place for the purposes of increasing user value and decreasing individual system maintenance costs. Since they already have their own independent systems, the agencies of the DoD will have to adapt to changes brought on by the GIG; this might be accomplished through incorporating an analysis of effective cost management, delivery time, delivery methods, risk analysis, system testing and recovery techniques. The GIG as a whole will also require evaluation for its survivability threshold, and for whether it has sufficient capacity to respond in a timely manner to potential attacks, failures, or accidents.

As they relate to the GIG system integration, emergent behaviors will appear when a number of single entities operate in an environment to form a unified, more complex system, resulting in a more convoluted system behavior. Emergent behaviors are often unpredictable or unanticipated, but can offer a new level of system evolution, creating dynamic relationships across different parts of the system, and providing feedback for the system's integrated components. While emergent behaviors can prove to be beneficial to system survivability, they also play a role with other nonfunctional global properties that relate to system performance, safety, security and reliability. Within the GIG system integration, emergent behaviors are the contributing factors that might offer alternative approaches, potentially resulting in lower production and communication costs that will be directly associated with the architectural design of this system.

When building a system of such magnitude, based on so many different sources, it is also important to consider the possibility that architectural mismatches may occur, which could keep the system from operating as intended. While emergent behaviors provide an alternative method for the system's components to communicate with each other, the architectural design of the system really makes the difference of whether or not the integrated systems will effectively be able to function collectively [Fisher 01]. Architectural inconsistencies are a critical issue in software design, and occur more often when the behavior analysis of systems are explored. When integration problems occur, component behavior and interaction and mismatch detection can be explored through behavioral analysis techniques [Uchitel 00]. When considering architectural mismatches, it is important to think about properties such as secure transactions, reasonable performance, methods of communication, compliance to real time needs, and the quality of service that will be affected by the overall integration of the systems that will compose the GIG. Architectural mismatches can occur for many reasons, for example premature assumptions based on nonexistent mechanisms or unsupported evidence. These have been known to lead to emergent behavior mishaps such as cascading failures. While cascading failures represent the worst-case scenario, they correspond to a series of progressive but continuous overpowering transitions whose net effects result in a system overload [Hines 05].

Unanticipated consequences occur as the direct result of some action that has taken place. As they pertain to the GIG, consequences are viewed as unanticipated when they represent events not originally considered potential outcomes. It is almost impossible to anticipate every possibility, but with certain precautionary actions, better analysis of problems beforehand could prevent a cascading failure from destroying an entire system. Unanticipated consequences also present positive outcomes. Whether positive or negative, they can force new perspectives on a situation.

This study provides information on maintaining a dependable and trustworthy system whose components are integrated from other infrastructures. The study also provides details on specific types of emergent behaviors that could occur during transformation of the various information systems to form the GIG, and the architectural mismatches and resulting consequences that could follow. Taking a look at two specific systems that will be integrated within the GIG, this study looks at how the systems will relate and possibly respond within the GIG system integration. This study concludes with a detailed look at unanticipated consequences as they relate to the GIG system integration and provides suggestions on how they could be prevented.

3.2 GIG System Integration

As stated previously, the systems of the DoD that will be integrated to compose the GIG will provide for the support, operation locations, and functional capabilities of all owned and leased communications, computing systems and services, software, data, and security services. As with any system, in order to effectively manage security tasks and system integration, it is important to address the trust and scalability characteristics of these tasks as they relate to concepts such as authentication, authorization, and access control. In order to build a quality-assured, trustworthy system, these qualities must be built into the original architectural design of the new system. To promote trustworthiness in relation to the GIG, it is important to consider, when integrating components, how emergent behaviors could lead to unanticipated consequences.

In order to make the GIG an effective, trustworthy system, the architects must consider how they can create collaborative working environments for the various integrated information systems [Schneider 99]. As unanticipated consequences occur, successfully integrating all the components of the system and understanding the role of trustworthiness in this integration will be a major challenge. Unanticipated consequences will be difficult to avoid, but certain system attributes should be considered: complexity, dynamics, intransparence, and ignorance and mistaken hypothesis.[9] These attributes are discussed further below.

Complexity reflects the various information systems and their associated components that will be used to compose the GIG architecture and the interconnections that will exist among these components. While the GIG system model will not include all the components of each information system, there is the danger of this occurring (i.e., system information overload) and resulting in an unanticipated consequence. Another important aspect to consider is that there will be many players with roles in the GIG system structure. While they will all be DoD related, they may not be interrelated in ways that will help identify and define the GIG system infrastructure, which entails who the key players will be, what authority they would have, and how the others will respond to the resulting change.

Dynamics may not directly relate to the GIG system because the governing structure of the GIG will not posses the ability to modify a system’s state independent of any authoritative user control—someone will be in control of all of the GIG system commands. It has yet to be decided if there will be one specific DoD agency in charge, or if there will be specific individuals (or divisions) of all the DoD agencies who not only contribute to the GIG system but will be deploying the system in their everyday combat procedures. However, unanticipated consequences could still occur as a result of improperly used system dynamics. There could be too many users assigned authority rights to the system, which could inadvertently cause conflicts within system. In addition, hostile parties could find a way to intrude upon the system and cause system complications by performing activities beyond the scope of any authorized person.

Intransparence refers to underlying elements within a system that cannot be seen but can directly affect quality operations of the system. As related to the GIG system, clear possibility exists that with so many different (but similar) information systems being combined, some hidden features may be left undiscovered. If exposed in an improper manner, these might affect the overall inter-working and quality of the system. The problems could involve a

[9] Healy, T. “The Unanticipated Consequences of Technology.” Santa Clara University: Markkula Center for Applied Ethics, 2006. Not publicly available.

minor system communication or could involve major information delivery or user access controls. For example, if warfighters are in combat and trying to access information using the GIG and their user access controls do not work, or if they attempt to deploy at a target and their missiles do not respond in time, this could create catastrophic, unanticipated consequences.

Ignorance and mistaken hypotheses are a possibility in almost every task we encounter in life, but in the case of the GIG system, there is a heightened possibility of occurrence. It is important to increase everyone's understanding—both of those who build and who will use the GIG—of the system and how it is projected to operate. This understanding must extend from the system's purpose and plan to future goals, so that accurate and precise hypotheses can be projected.

3.3 Emergent Behaviors

Emergent behaviors tend to appear once a number of single entities begin to operate in a cohesive environment forming a unified, complex system that in the end will result in a convoluted system behavior. Within the GIG system integration, emergent behaviors will be the contributing factors that offer alternative approaches to lowering the production and communication costs associated with the architectural design of this system. Emergent behaviors not only further system survivability, but they contribute to satisfying mission requirements and producing an overall quality structured system [Fisher 01]. There are many different types of emergent behaviors to consider when integrating a system, but for a system with the size and operational capacity of the GIG, it is important to consider some of the more destructive types of emergent behaviors—cascading failures, deadlocks, data communication errors, and distributed denial of service attacks—that if not managed carefully, could result in fatal system overloads.

Cascading Failures: A cascading failure is a series of progressively continuous events that tend to have overpowering transition net effects that can result in a complete system overload [Hines 05]. Cascading failures are common in systems with interconnected parts where the service provided depends on successful operation of the preceding part. If there is failure with the preceding part, this can trigger failure for the successive parts. For example, with the GIG, if a warfighter is in battle and needs to execute some control commands, he may attempt to log on to the system, not knowing of pre-existing conditions. He uses his access code and gains access into the system, but instead of the system directing him to where he needs to go, the system is in a constant loop and continuously takes him back to the main login page. With this type of system malfunction, the warfighter will not be able to execute the control commands that he needs, which in turn could lead to some very serious consequences occurring due to the login system malfunction. One aspect of the GIG that may help to reduce the risk of cascading failures is its planned operation in a near real-time state. This would require that someone continuously monitor the system and could prevent incidents like the one described above from taking place.

Deadlock: A deadlock is a situation where two or more competing actions are waiting for the other to finish executing their actions, but neither of them ever does. In a deadlock situation, both executions cannot be satisfied at once. This causes the two or more processes that are waiting to compete amongst themselves in a circular chain as each waits for the other to complete its cycle. For example, think about the problems that would occur if the GIG only let one division of DoD access the GIG at a time—if the Army is using the system at the present time, but the Air Force needs to have access. Unaware that each division may have a specific time to use the system, the Air Force could be in mid-battle, not knowing what the Army is using the system for, or for how long. The Air Force cannot access the system and therefore is in a deadlock situation, that is, it cannot access the GIG, and it cannot execute the commands needed for battle. While there is no general solution for complete deadlock prevention, each type of deadlock can be anticipated and specifically prevented beforehand. In the situation described above, the deadlock could have been avoided if certain information about the processes of using the system resources of the GIG were known prior to system allocation and alternative results had been considered.

Data Communication Errors: Data communication errors can occur in a system when data is sent syntactically correct through the system, but may have different meanings in various places and therefore does not reach its destination with the proper message. As with the GIG, this could happen in a number of ways since the GIG will be composed of a number of different information systems that have their own methods of communicating internally within their organizations. For example, if the Air Force sends a command through the GIG that originates from the Theater Battle Management Control System (explained in greater detail in Section 3.5), but then another component of the GIG reads the system command and finds that the language has not yet been changed to suit the needs of the new system specifications, this could cause defaults in the system data communication errors.

Distributed Denial-of-Service Attacks: This type of attack occurs when a multitude of compromised system attacks occur on a single target, causing a denial of service for the users of the targeted system. This ultimately causes the system to flood due to all the excessive incoming messages, ending with the system completely shutting down due to denial of service. Unlike the previous examples that relate to the GIG system usage, this type of problem will not arise from faulty communication channels but from a direct attack on the GIG network. This attack will not only cause a loss of service to users, but will consume all of the system's bandwidth, making the system susceptible to other intrusion attacks.

3.4 Architectural Mismatches

Even with algorithms in place to deflect the negative influence that emergent behaviors can exhibit, it is still possible in a system of the GIG's magnitude for architectural mismatches to occur that prevent the system from operating as intended. For example, when AOL added virus protection, spam filters, and privacy features to its service plan, it had to make sure that its services would complement the security software of systems that could already be in place on a user's system (e.g., Norton Anti-Virus software), and would not disrupt the computer's pre-existing stable environment. The additional services offered by AOL enhance the security features a user already has in place and do not overrun them. This same concept can be applied to the GIG. When combining the information systems of the various DoD agencies into one cohesive structure, neither system is to overshadow the other, but to complement the other to maintain a sound operational system. Since each system being integrated has unique parts, architectural mismatches may occur. However, these should not present a true threat to the overall system environment if proper system recovery plans are considered with numerous assumptions in place to cover a wide-range of possible consequences. The idea of architectural mismatch is associated with the idea of assumptions, which may lead to unanticipated consequences that prevent the system from performing properly.

In today’s fast-paced software-creating world, architectural inconsistencies are common, and present a critical flaw in the design of software. This could be a direct result of system vulnerabilities, software flaws, or improper testing techniques; however, to explore architectural mismatches more directly, it may be helpful to use component behavior analysis techniques that deal with component behavior and interaction, and to use mismatch detection schemas [Uchitel 00]. Architecture mismatches can often be found within flawed software, and difficulties may present themselves when

- communication is attempted across secure boundaries
- reasonable performance standards are not met
- methods of communication are inconsistent
- conflicts arise due to the system’s not being compliant to fit real-time needs
- the quality of service that is rendered from the overall system capabilities is not sufficient

There is not one simple way that architectural mismatches occur—they can come from any direction, for any reason, be singular or grouped, and can occur multiple times. However, they are more common to occur with system design flaws, system capacity overloads, user interference, and system crashes (which could be a result of hostile party interference). When building the GIG system, the architects must consider the most effective way to combine the different systems and create new styles for a shared infrastructure. The shared architecture must be incorporated into every environment of the GIG, but would function as a support service for the architectural design [Garlan 95]. For this shared infrastructure, the architects might consider modifications in vocabulary, visualization, and a new set of analysis tools that will represent the new environment. Another useful idea for the architects to consider is to pick systems and elements within the systems that seem to have a promise for working well together within the cohesive environment. Architectural mismatches would of course occur in this instance if elements are chosen from systems that don’t have any similar characteristics. Forcing the systems to work together could also create an architectural mismatch. Therefore a shared infrastructure plan seems to be the most effective way to detect when and how mismatches occur.

Can architectural mismatches be avoided? There are no simple answers. The harsh reality is that when you build any system, whether from scratch, or in this case from software integration of other established systems, the transition will not be easy and flawless. There will be integration problems either from an excessive amount of code being used or poor overall performance of the system. There will also be a need to modify external packages and tools: understand the nature of the components and how they work, reinvent existing functions, redesign data communication mechanisms, restructure the global architecture structure, and eliminate unnecessarily complicated tools that can minimize some of the errors that may occur with the construction process [Garlan 95]. In reality, there may not be a definite way to avoid architectural mismatches, but if they do occur there must be designated ways to fix the resulting problems; the GIG won't simply stop due to an architectural mismatch. Methods have been developed that architects can use to build components that are suited to fit a large system. These methods could provide new mechanisms and tools to use with the systems functions and hopefully minimize architectural mismatches. Use of such methods would yield a way to make the architectural assumptions more clear and well defined and provide techniques for reducing architectural mismatches. Should mismatches occur, a recommended source of guidance will need to be in place to help rectify problems as they arise.

Although the difficulty of avoiding architectural mismatches has been discussed along with ways to minimize their occurrence, it is also important to consider the worst-case scenario should mismatches occur repeatedly. There are many system failure functions to consider; as discussed earlier, emergent behaviors have strong effects with system deadlocks, data communication errors, distributed denial-of-service attacks, and cascading failures. While each of these presents trouble in a system, it is the cascading failure that can present severe damage. A cascading failure is a series of progressive, continuous events that have over-powering transitions whose net effect typically results in complete system overload [Hines 05]. This is not an immediately recognizable situation. Events must occur over and over again before a cascading failure transpires; if effective functional reactive control recovery plans are in place, the "same" problems with architectural mismatches won't keep repeating. Once a cascading failure takes place, as with any system malfunction, the solution must be provided within the system recovery plans to get the system back up and running as it was prior to the overload. While many cascading failures are propagated by repetitive system defaults, if the GIG system architects plan early on to eliminate network violations and can capture the global control of the problems as they present themselves, the stress on the network won't cause cascading failures. This avoidance will clearly show the advantages of long-term, advanced planning.

3.5 Systems Being Integrated to Form the GIG

The discussion of this study thus far has covered

- the GIG system integration—what will be included, and how to define trust within the resulting relationships
- emergent behaviors—what they are, and how their actions can directly relate to the GIG system survivability
- and architectural mismatches—what they are, how to prevent them from occurring, and the worst-case scenario that results when they occur multiple times

In a more detailed look at the GIG system integration, this study examines two of the systems that will actually be included within the integration and discuss the actions that will take place. The systems that will be discussed are the Theater Battle Management Control System (TBMCS), a U.S. Air Force led initiative, and the Advanced Field Artillery Tactical Data System (AFATDS), a U.S. Army and Marine led initiative.

3.5.1 Theater Battle Management Control System

The Theater Battle Management Control System (TBMCS) implements interoperable functionality with other command, control, communication, computer and intelligence (C4I) systems involved in theater air warfare. It provides the Combat Air Forces (CAF) and the Joint/Combined Forces with automated and integrated capabilities to plan and execute the air battle plan for operations and intelligence personnel at the force and unit levels [FAS 99]. The TBMCS can be tailored to suit the needs of both large- and small-scale operations for varying intensities of warfare. However, its primary goal is to provide the air commander with the means to plan, direct, and control all theater air operations for the overall support of command objectives and to coordinate with the ground and maritime elements that are engaged in the same operation [GlobalSecurity.org 05]. The TBMCS system also supports peacetime training and daily operations as well as timely reactions to contingency plans.

Providing a single point of access to real- and near real-time information for the systems users, TBMCS was manufactured by Lockheed Martin Mission Systems, and comprises a 27-workstation host system with remote access controls located throughout. Although the system was initiated by the U.S. Air Force, Lockheed Martin created it to function for the training and implementation strategies of all U.S. forces. Presently used as the DoD primary command and control (C2) system for air battle management, the TBMCS is equipped to handle 1500 missions and 3000 daily sorties [Lockheed Martin 05].

3.5.2 Advanced Field Artillery Tactical Data System

Providing flexibility to manage attacks on preplanned and time-sensitive targets, the Advanced Field Artillery Tactical Data System (AFATDS) acts as a fire support server to LAN-based and Tactical Internet-based clients [Raytheon 05]. AFATDS provides a net-centric battle command to coordinate land and air indirect fire systems in support of Army and Marine maneuver operations. AFATDS also processes fire mission commands to coordinate and optimize the use of all fire-support assets, support planning, and current operations, including mortars, field artillery, cannon, missile, attack helicopters, air support, and naval gunfire [FAS 98].

During battle, AFATDS will provide users with up-to-date battlefield information, target analysis, and unit status, while also coordinating target damage assessments with sensory-related applications. Processing over 400 requests per hour, in battle, AFATDS provides fire support by ensuring that weapons fire the ideal munitions at their targets with the highest tactical payoff [GlobalSecurity.org 05]. In addition to fire support efforts, AFATDS meets field artillery specifications by managing critical resources, supporting personnel assignments, collecting and forwarding intelligence information, and controlling logistical functions.

3.5.3 The Systems Relationship to GIG System Integration

AFATDS is a multi-service automated C2 system that is set up to not only function as a fire support operation, but will interoperate with the fire support C2 systems for Germany, the United Kingdom, and France [Boutelle 96]. By integrating the fire support systems via a distributed processing system, AFATDS allows missions to be planned and carried out in less time than they would through single systems. Also designed to respond efficiently to emerging needs, AFATDS communicates over a local area network between command posts while transmitting information on a single-channeled ground and airborne radio system. Since it was designed to be more interoperable with other systems than the TBMCS, the specifics of AFATDS could prove to fit the specs of the GIG with greater ease, and fewer architectural mismatches, than that of its “air-controlled” counterpart. The AFATDS is directly aligned with GIG system specifications, and with what the DoD unified system will be designed to include. It is therefore necessary to consider the 27 fire support functions that relate to AFATDS operational needs; these can be categorized as fire support execution, fire support planning, movement control, field artillery mission support, and field artillery fire direction operations.

TBMCS integrates all C2 control centers, and provides a comprehensive air battle planning and mission execution approach for all involved services. Unlike AFATDS, TBMCS cannot be integrated with different systems with relative ease because the system specs relate directly to airborne initiatives, while those of AFATDS have a more diverse range of field tactics. When integrating the TBMCS to fit the specifications of the GIG system, the architects will have to consider the most effective ways to include the TBMCS system air mission operational strategies and make them complement those of others that target both the land and water missions. With both systems concentrating on timely deliverables of information to the systems users, the GIG architects must focus on coordinating which system commands controls will be used, when they will be deployed, and in what fashion to effectively minimize any specific architecture mismatches that might occur. As with AFATDS and in line with GIG system specifications, architects must consider the TBMCS software categories that will have a direct functional impact on the interworking of the GIG system: planning, executing management, resource management, reporting and analysis, and common tools used.

3.6 Unanticipated Consequences and the GIG

The DoD has formally made its decision; the GIG system formation will take place. For the integration to occur successfully, the agencies must choose

- which systems will perform most effectively within the GIG

- who will be the authorized users of the system
- where the main database will be stored,
- an effective system recovery plan with alternative solutions to certain problems that can be foreseen
- a testing-phase release date

Many of the DoD agencies have their own information systems, and as discussed earlier, each agency system may have quality attributes that could fit into the framework specifications of the GIG. However, each quality might not appear holistically in the system because of compatibility, survivability, dependability, and system trustworthiness issues.

When looking at the overall GIG system scope and methodology, and considering all of the benefits that the new system will bring, it is also important to reflect on some of the negative effects and their related consequences that may follow. In almost all cases, actions have effects that are unanticipated, but it is the consequences of the events that are not foreseen and dealt with in advance that could have the most perverse effects on the GIG. Anticipated consequences are those that are intended and desired, common or probable, but unanticipated consequences can be both desirable and undesirable. [10]

3.6.1 Considerations for the GIG and Unanticipated Consequences

Change will always take place, whether it is through human or technological intervention, or simply evolution, it is impossible to escape change and the probable unanticipated consequences that it brings. When discussing unanticipated consequences, it is valuable to consider the basic four features of any system: (1) complexity (2) dynamics (3) intransparence, and (4) ignorance and mistaken hypothesis.[11] This relates to points discussed earlier:

- Complexity reflects the information systems of the GIG as it relates to the components and interconnections that will occur within the system.
- Dynamics deals with the governing structure of the GIG and who will possess authoritative control.
- Intransparence identifies elements within a system that are hidden from the user eye and can affect the overall quality of the systems operations.
- Ignorance and mistaken hypotheses have a possibility of occurring due to unforeseen events that can affect the interworking of the systems operations.

3.6.2 Preventing Unanticipated Consequences

In the end it is impossible to anticipate everything. Building the GIG system is very complex, and while the architects might have plans mapped out to deal with destructive situations, including alternatives if the original plans don’t work, they cannot predict every possible occurrence. Therefore, unanticipated consequences should never be ruled out; however, if the

[10] Healy, T. “The Unanticipated Consequences of Technology.” Santa Clara University: Markkula Center for Applied Ethics, 2006. Not publicly available.

[11] Ibid.

parties involved would proactively work towards a clearer understanding of the system and require the system to be more effective in its operations, then the uncertainty could possibly be reduced. Since unanticipated consequences are unavoidable, it is important that the GIG system has proper recovery techniques to maintain the system’s trustworthy principles and its dependability so it can survive even the worst possible threat.

In specifically focusing on the TBMCS and AFATDS systems in relation to previously discussed emergent behavior problems that could plague the GIG system, it is important to consider possible unanticipated consequences and what could be done to prevent them. Once integrated to fit the specs of the GIG system, the applications of the TBMCS and AFATDS will no longer have individual identities, but will operate through a shared infrastructure. While they may have some shared similarities in operational demands (the reasons that these systems were among those chosen to be integrated for the GIG), the systems are not directly compatible and problems such as cascading failures, deadlock, data communication errors, and distributed denial-of-service attacks could occur.

These emergent behaviors could have a negative influence on the use of the new system due to unexpected events that surface. If there is no effective system recovery plan to combat the problems with emergent behaviors some severe system complications could result. In order to minimize unanticipated events, the architects of the GIG should thoroughly research the history of both the TBMCS and AFATDS. The agencies that control the operational components of the systems should have accurate resources that specify what type of mischievous activities, system malfunctions, and other related flaws have occurred over the history of the systems existence. This information should also enable the architects to frame their design of the GIG to reduce the reoccurrence of past problems.

3.7 Conclusion

The DoD will implement the GIG as its net-centric information system that will operate in a global context providing authorized users with a secure and interconnected information environment, while meeting near real-time needs of both the warfighter and the business user. The GIG will not be created from a simple idea, leading into some big plan that will affect the entire DoD. Instead several of the compatible information systems that are already effectively providing services to the agencies of the DoD will be integrated to compose the GIG. Providing support, operation locations, and functioning capabilities as they relate to national security, the GIG will include all owned and leased communications, computing systems and services, software, data, and security services. Effective management of system costs, delivery time, delivery methods, risk analysis, and system testing and recovery techniques will be imperative. The GIG will need to be tested against its survivability threshold to make sure it has sufficient system capacity to respond in a timely manner to attacks, failure, and accidents should they occur.

Emergent behaviors will appear within the GIG system integration as a number of single entities operate in a unified environment forming a more complex system that could result in a more convoluted system behavior. While often unanticipated, emergent behaviors offer a new idea for system evolution that creates casual relationships across various parts of a system while providing feedback for the integrated system components. When studying system integration within the GIG, it is the negative effects of emergent behaviors that should be considered as offering alternative approaches to problems while also potentially contributing to lower production and communication costs that relate to the design and functionality of the system.

Architectural mismatches are another important factor to consider when building the GIG because with so many various systems being implemented into one, there could be architectural inconsistencies that create flaws in the system and will cause the system to not function dependably. When integration problems occur within the GIG, the component behavior, interaction, and mismatch detections can be explored through behavioral analysis techniques. This could also yield specific breakdowns for flaws that might occur with the systems secure transaction mechanisms, methods of communication, and overall compliance to system service needs. Also, when architecture mismatches occur repeatedly and of the same context, often times they could lead to cascading failures (which is the worst-case scenario) which could have an end result of total system failure or capacity overload.

Also discussed in this study are two of the systems that will compose the GIG: Theater Battle Management Control System (TBMCS) and Advanced Field Artillery Tactical Data System (AFATDS). TBMCS provides U.S. Forces with automated and integrated capabilities to develop and execute the air battle plan for operations and intelligence personnel at the force and unit levels. TBMCS integrates all command and control centers, and provides a comprehensive air-battle-planning and mission-execution approach for all involved services. On the other hand, AFATDS provides a net-centric battle command to coordinate land and air indirect fire systems in support of Army and Marine maneuver operations. Enabling missions to be planned and carried out in less time than previous systems, AFATDS integrates the fire support systems via a distributed processing system, while communication is processed over a local area network.

Unanticipated consequences arise from situations occurring as the direct result of some action that has taken place (e.g., conflicts among implemented systems composing the GIG) that did not transpire according to the original plan. Sometimes the unanticipated consequences shed light on a problem so that it can be corrected and thereby prevent a more serious future negative event Proper system recovery plans are crucial for rectifying destructive situations so that problems can be eliminated and hopefully never transpire again.

4 IPv6 Network Addressing: Architectural Properties and Operational Survivability Effects

4.1 Introduction

The employment of Internet Protocol version 6 (IPv6) is one of the key Department of Defense (DoD) mandates for next-generation military data networks, including the Global Information Grid (GIG). Operational capabilities such as Joint Battle Management Command and Control (JBMC2) will in large part depend on these networks, so it is important to analyze and understand the implications of using the technologies underlying them. This understanding is especially critical for technologies such as Internet Protocol (IP), which provide the foundations for network architecture and the universal basis for network operations. The eventual replacement of IPv4 with IPv6 will effect a fundamental change in network architecture and operations. This is likely to result in significant changes to system attributes, including survivability. Understanding the architectural and operational features of IPv6 will allow greater understanding of the survivability of IPv6-based networks. This in turn will allow greater understanding of the survivability of capabilities such as JBMC2 and the GIG as a whole.

The concept of “system survivability” will have different meanings in different contexts. In this study, system survivability is understood to be a composite of other system attributes. The effect of a technology such as IPv6 on survivability will be understood in terms of its effects on some subset of these constituent attributes. The goal of this study is first to understand what system attributes are relevant to system survivability and second to identify which of these attributes are relevant when discussing the architectural and operational effects of IPv6. IPv6 is actually a collection of specifications which make a number of changes to the IP, and this study analyzes only a single, albeit fundamental, aspect of IPv6, namely network addressing architecture. As will be seen, the aspect of network survivability most affected by network addressing architecture is the attribute of operational control (defined in more detail below), which is operational, as opposed to architectural, in focus.

System attributes that are operational in focus are particularly relevant to networks supporting operational concepts such as JBMC2. The reason for this is twofold. First, networks used to support joint missions will be highly complex systems of systems (SoS), in both horizontal and vertical respects. In a horizontal sense, a network supporting JBMC2 will involve a collection of systems not managed by a single entity but by separate DoD organizations, non-DoD agencies, Services, and separate divisions within the Services [DARPA 01]. For instance, one joint doctrine publication describes thirty component agencies involved in the Close Air Support function [JP 3-09 03]. From a network management perspective, these agencies' systems will likely involve independently administered autonomous systems (AS's). The availability of these systems will largely depend on the ability of these AS's to coordinate routing and address allocation policies, inter-domain quality of service (QoS), and incident response planning. To resolve problems requiring horizontal coordination, shared situational awareness among autonomous network operators is essential [USAF 01].

In a vertical sense, IP-based networks of any value involve multiple overlays of systems with critical dependencies among them. Vertical dependencies involve both upward and downward flows of information and control (Figure 2). For instance, resolving routing issues at OSI Layer 3 requires understanding and control of the underlying Layer-2 infrastructure. Similarly, most application-layer systems require the availability of the underlying Domain Name System (DNS) infrastructure. In contrast, the provisioning and management of Layer-2 and Layer-3 configurations may require understanding and control of higher layer systems (such as database and staging platforms or a public key infrastructure [PKI]). To resolve problems requiring vertical coordination, centralized control by an agent able to make sense of and resolve inter-layer dependencies is essential [USAF 01].

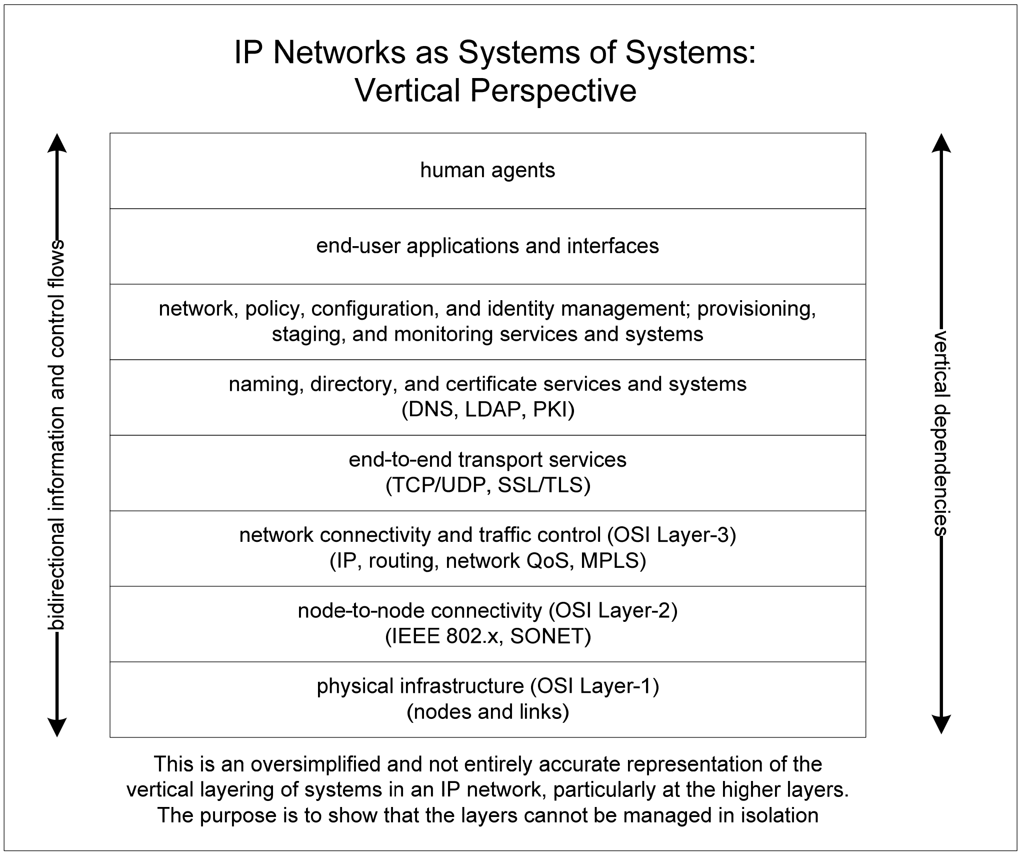

IP Networks as Systems of Systems:
Vertical Perspective
bidirectional information and control flows
vertical dependencies
human agents
end-user applications and interfaces
network, policy, configuration, and identity management; provisioning, staging, and monitoring services and systems
naming, directory, and certificate services and systems (DNS, LDAP, PKI)
end-to-end transport services (TCP/UDP, SSL/TLS)
network connectivity and traffic control (OSI Layer-3) (IP, routing, network QoS, MPLS)
node-to-node connectivity (OSI Layer-2) (IEEE 802.x, SONET)
physical infrastructure (OSI Layer-1) (nodes and links)
This is an oversimplified and not entirely accurate representation of the vertical layering of systems in an IP network, particularly at the higher layers. The purpose is to show that the layers cannot be managed in isolation

Figure 2: Vertical Perspective IP Networks as Systems of Systems

The second reason operationally oriented system attributes are especially relevant to networks supporting JBMC2 is that the systems will be operating in highly dynamic environments characterized by uncertainty. Because of unpredictable conditions and the potentially rapid pace of contingencies in the JBMC2 environment, relying solely on architectural components of network survivability (building in network redundancy and diversity) is insufficient for ensuring the overall survivability of the system. The level of complexity associated with systems of systems, as well as the time-sensitivity and indeterminism of their operating environments, suggests that coordination among human agents and positive control of the network by human agents—that is, operationally focused attributes—are critical to system survivability. The following section will describe these attributes in more detail and analyze how they are affected by network addressing architecture.

4.2 Network Survivability and Network Addressing Architecture

4.2.1 Defining System Survivability

Definitions of system survivability vary considerably. Some definitions are undeveloped or one-dimensional, some are relatively narrow and specific, and others are more general or comprehensive. In the undeveloped category, survivability has been associated with system protection and redundancy [USAF 01] as well as heterogeneity in system elements [Zhang 01]. More developed notions of survivability involve multi-dimensional models. One such model describes survivability as a set of alternate "service levels" interrelated by a state transition diagram and uses probabilistic measures to determine "survivability specifications" [Knight 03]. Finally, there are broader and less rigorous definitions describing survivability as a non-functional attribute of a system comprising a set of capabilities [DARPA 01, Mead 00]. For the present purpose of understanding the survivability effects of a network technology such as IPv6, the decomposition of survivability into specific capabilities will prove most practical and appropriate analytical approach.

The SEI concept of network survivability relies on a system's capabilities to deliver essential services by means of four general properties [Mead 00]:

1. resistance to attacks
2. recognition of attacks and damage
3. recovery of essential and full services after attack
4. adaptation and evolution to reduce effectiveness of future attacks

These "3R+A" properties in turn are achieved through multiple system capabilities. For instance, the "resistance" property may be strengthened by the employment of authentication and access control mechanisms, while the "recognition" property may be enhanced through instituting monitoring procedures. While the 3R+A capabilities catalogued in Mead are architectural in focus, a more comprehensive description should incorporate operational focus as well [Mead 00]. For instance, the recognition strategy of intrusion detection requires an architecture for deployment of the sensor, analyzer, and reporting components of the intrusion detection system (IDS), including organizational processes integrating the IDS into the overall system. However, there are requisite operational capabilities such as situational awareness which allow the IDS to be employed successfully. For instance, the value of an IDS is greatly enhanced—or, arguably, realized—by human agents who have the experience, contextual knowledge, and situational awareness to distinguish true alarms from false positives quickly and efficiently. Since Mead presents an architectural analysis method, it makes sense that the 3R+A strategies are architectural in description [Mead 00]. Nevertheless, it should be understood that the presence of 3R+A capabilities in a network is not a stand-alone indicator that the 3R+A properties have been achieved.

In its study on the survivability of GIG systems, the Defense Advanced Research Projects Agency (DARPA) complements architectural properties with operational capabilities and considers network survivability to be a function of both. The architectural elements include

1. connectivity, or the resiliency of communication paths through nodes and links
2. security, or the ability to control network resources available to authorized and unauthorized users
3. network management, or the ability to maintain and configure network nodes
4. directory services, or the provision of “usability” features on a network, including name and addressing services, service registration, and user authentication and authorization

A notable aspect of DARPA’s proposed architectural elements is that three of them, security, network management, and directory services, emphasize the operational benefits of architecture, which are given in terms of capabilities for operational control of network systems. The security element includes two control mechanisms: the “ability to assert positive control over the resources of authorized network users” and the “ability to deal with unauthorized activity” [DARPA 01]. Ideally, the security element includes “cognizance of all activity on the network and...the intelligence to curtail any traffic that does not belong.” The network management element involves control over nodes, applications, and traffic flows. Finally, the “usability” functions included in the directory services element, such as key management and user authentication and authorization, are in fact network control mechanisms [DARPA 01]. In this respect, directory services are not easily distinguishable from the security element. Operational survivability is further emphasized in the DARPA model’s application of the disciplines of security and network management to three operational domains [DARPA 01]:

1. configuration control, or setup and maintenance of network services
2. network defense, or managing mechanisms protecting the network
3. managing degraded service, or managing mechanisms adding dynamic resiliency to the network

4.2.2 Defining Operational Control

The overall system survivability attribute emphasized throughout DARPA’s model is operational control, and it is in this regard that the fundamental change in network addressing architecture offered by IPv6 will make the biggest difference. However, before discussing in detail the relevance of network addressing architecture to operational control, it is necessary to present a brief analysis of what is meant by the term. Operational control can be viewed as comprising a continuous cycle of three elements:

1. cognizance, or awareness of the facts and their context in a given situation
2. analysis, or the informed decision-making process in which human agents formulate a course of action
3. execution capability, or the ability to effect changes based on the decided course of action (see Figure 3)

An example breaking down a hypothetical incident response action into operational control elements serves as an illustration. Say that a service-provider network is subject to a distributed denial-of-service attack. To deal with the attack, analysts first need cognizance of the attack traffic characteristics, for example, entry points, traffic rates, packet characteristics, and target IP addresses. Once these facts and their context are known and verified, analysis can proceed to conduct triage, determine the mitigation strategy, and construct a candidate configuration for network elements (e.g., creating sinkholes on various routers in the network). Finally, execution capability (e.g., the ability to reach and connect to the management interface of relevant devices and the authorization to make the intended changes) will allow analysts to implement the configuration in a timely and intended manner. Another cycle follows, as the analysts need to understand the new situation to conduct further analysis and execute further changes if necessary.

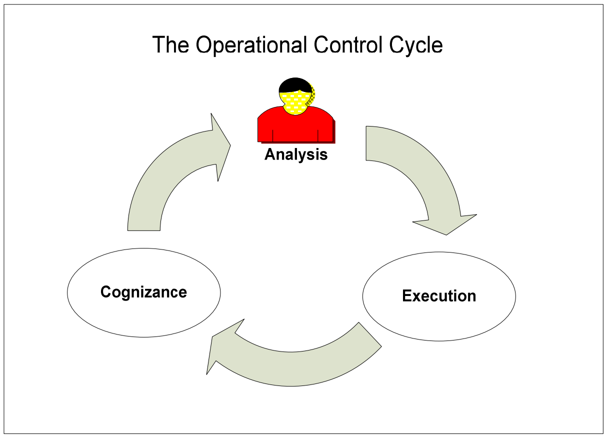


Figure 3: The Operational Control Cycle

4.2.3 Network Addressing and Operational Control

Addressing is critical to operational control because of its two fundamental network roles: identification and forwarding (Figure 4). Full network addresses are configured on devices in order to identify both hosts and network services, while network address prefixes are configured on network devices to identify subnetworks and forwarding paths, both of which establish a picture of network topology. Full network addresses are also present in packet headers, and here they represent destinations (i.e., forwarding instructions) to network devices. As identifiers, network addresses are obviously critical to the cognizance element of operational control. Source and destination addresses in packets provide two elements of the “5-tuple” used in characterizing and analyzing network traffic (e.g., in network flow data). Addresses also provide cognizance through their inherent role as communication endpoints and execution capability through their use as descriptors for hosts and services. As communication endpoints, addresses enable network management devices to identify control traffic with target hosts. As descriptors, addresses enable both implementations of network control policy (e.g., in firewall and other device configurations using address-based access controls) and parameters for network traffic management (e.g., in routing or QoS policy configurations). In terms of the DARPA model, the cognizance and execution capabilities enabled by addressing are central to almost every survivability component in the three operational domains, particularly in the configuration control domain. In terms of the SEI model, the cognizance provided by network addressing is *sine qua non* for achieving the architectural property of recognition. Finally, the structure and representation of network addressing may have an impact on the analysis element of operational control. Analysis is defined here as the exclusive domain of the human agent. The human analyst accepts input conditioned by cognizance and produces output realized by execution capability. Addresses must have human-readable forms, and the manner in which they are represented and bear (or do not bear) information affects the decision-making process of the human analyst.

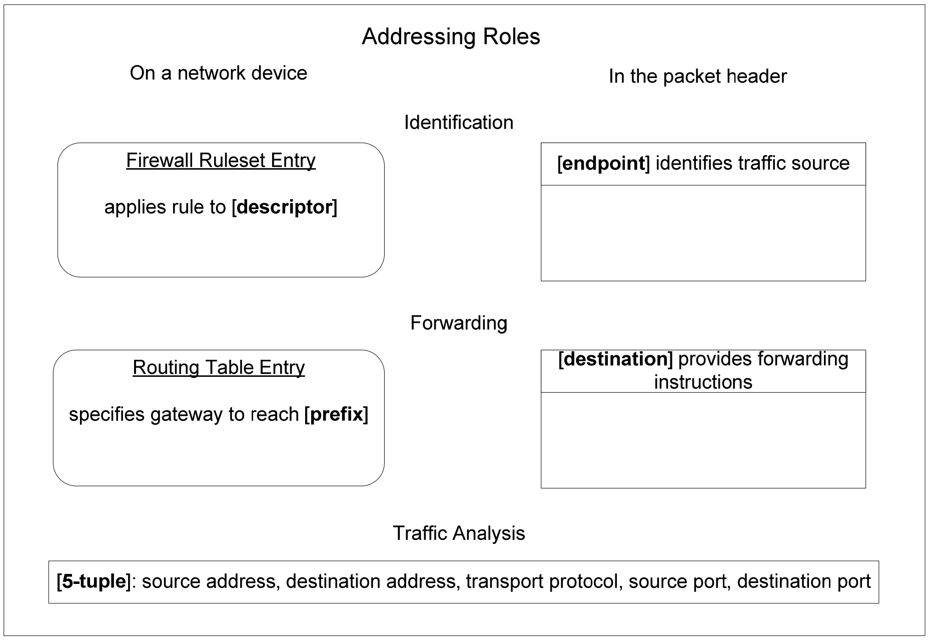

Addressing Roles
On a network device
In the packet header
Identification
Firewall Ruleset Entry
applies rule to [descriptor]
[endpoint] identifies traffic source
Forwarding
Routing Table Entry
specifies gateway to reach [prefix]
[destination] provides forwarding instructions
Traffic Analysis
[5-tuple]: source address, destination address, transport protocol, source port, destination port

Figure 4: Addressing Roles

The identification function of network addressing has resulted in one of the greatest problems with operational control of networks: the misplaced reliance on addresses as tokens for authentication. The problem is with the distinction between what will here be described as binding and association. Both properties are important inputs to cognizance and execution capability, but when confused they often result in reduced operational control. Authentication is a system's act of binding a principal (e.g., a human user) to a process in order to provide a level of assurance that the process is what it claims to be (i.e., bound to a legitimate principal). The problem is that network addresses are labels which only indicate superficial association (i.e., they *imply* that a principal is behind a process), which provides a far lower level of assurance than binding through authentication (Figure 5). Network addresses are a mainstay of operational control, but if misinterpreted they may also be a source of a false sense of operational control. Architecturally, the trust accorded to the association property has resulted in non-authenticating protocols on the local network such as address resolution protocol (ARP) and vulnerability to network-layer attacks involving spoofed source addresses. Finally, IPv4 as currently deployed requires mechanisms such as network address translation (NAT) and dynamic addressing of hosts, both of which may result in reduced operational control due to lack of transparency and persistence in host addresses. All of these issues contribute to one of the fundamental problems with operational control of networks: the "entry-point anonymity" problem, or the inability of a network system to bind principals to network addresses [Lipson 02].

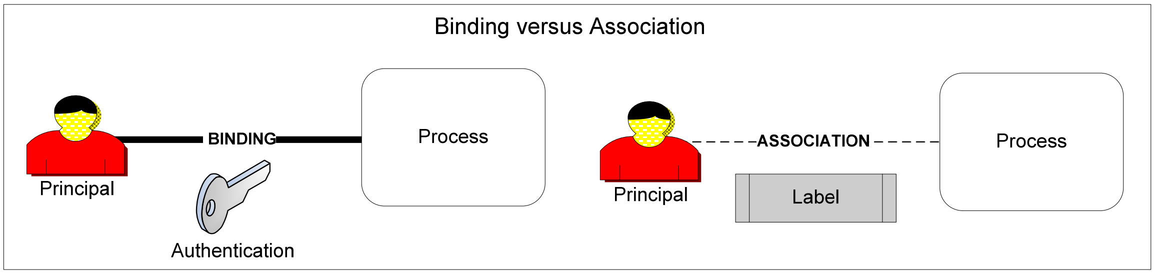


Figure 5: Binding versus Association

The remainder of this study examines the aspects of the IPv6 specification related to network addressing and discusses the impact of these architectural changes in terms of the elements of operational control, namely cognizance, analysis and execution capability. Some examples illustrating the impact on these elements will be drawn from the architectural elements and operational components of survivability put forth in the DARPA model. Although the focus of this study is to demonstrate the effects of IPv6 addressing on operational survivability capabilities, impacts on architectural attributes will be discussed where necessary.

4.3 Overview of IPv6 Network Addressing Architecture

Network addressing architecture is only one, albeit fundamental, part of the IPv6 specification. The following technological overview is not comprehensive and omits a number of key elements of IPv6 (e.g., the significant change in IP header architecture), each of which merits a similar survivability analysis. Furthermore, the overview is not a tutorial. It explains high-level features of network addressing under IPv6 that are relevant to identifying broad architectural properties, which in turn provide the framework for the survivability analysis. An excellent, although not completely up-to-date, introduction to the IPv6 specification is available for free online and provides a valuable reference [Kozierok 05].

4.3.1 128-Bit Addressing

The most widely known change in IPv6 is its expansion from 32 to 128 addressing bits. The designers intended the dramatic increase not only to supply more-than-sufficient addressing capacity for the foreseeable future, but also to make possible a number of other valuable properties, including structural flexibility and ease of management. The sheer size of 128 bits allows the architecture to "waste" considerable amounts of address space for these purposes. For example, the host portion of the IPv6 address has been predefined as 64 bits in length. The effect is that every network segment, including point-to-point links, is able to address in excess of 18 quintillion hosts, most of which is obviously unused. Furthermore, the ability to predefine these boundaries in the 128 bits of the addressing field removes the need to interpret the bits variably through the use of techniques such as bit-masking to identify the network and host portions of a unicast address (a key method for address conservation purposes under IPv4). Another effect of plentiful addressing is that it obviates the need for such widely deployed address conservation techniques as NAT, a change that has important architectural implications (see Section 4.3.7) [Egevang 94].

4.3.2 Structure and Meaning in Addresses

The use of globally predefined boundaries in IPv6 address space imposes a static structure on an address, which in turn provides inherent meaning for the bounded portions of the address. Predefined structure results in a number of useful capabilities, including the alignment of addressing with network topology (discussed further below) as well as the ability to predefine a number of address types, including special-use addresses, some of which are critical to IPv6 network functionality. In contrast, the absence of predefined structure within these bounded portions allows for flexibility in their use by the responsible organizations. For example, within the top 48 bits (the "global routing prefix") of unicast address space, Internet Registries have defined, through policy, allocation sizes which have been designed to promote the correspondence of addressing and topology and prudent usage. The current minimum allocation from the American Registry of Internet Numbers (ARIN) to an Internet Service Provider (ISP), for instance, is a 32-bit block of space (in classless inter-domain routing notation [CDR], a "/32"), which under the current global unicast format allows the provider to allocate 80 bits of addressing for each of roughly 64,000 end-user organizations [ARIN 02]. These allocations will allow for the first 32 bits of a given unicast address to "mean" that topologically, the address is exclusive to that provider.

4.3.3 Representation of Addresses

Operationally, it is necessary to represent a human-readable form of network addresses. IPv4 solves this problem by using dotted-decimal notation, in which a 32-bit address is represented by four 8-bit sections ("octets") written in decimal and delimited by periods. To represent 128 bits in the same manner (sixteen octets) is more unwieldy and negates the main advantages of using dotted-decimal representation: readability, recognizability, ease of configuration, and (to an extent) memorizability. IPv6 address representation is based on eight 16-bit sections ("words") written in hexadecimal and delimited by colons. Hexadecimal is used to make the address as compact as possible as well as to facilitate conversion with binary. Nevertheless, an eight-word address in hexadecimal is not easy for a human agent to manage, and additional techniques are used to compress the address further for the sake of readability and ease of configuration. Prefixes are represented in a manner similar to CIDR notation, in which the number of significant routing bits follows a slash appending the prefix [Fuller 93].

4.3.4 Addressing Types

There are currently three basic addressing types specified in IPv6. Each addressing type represents a different forwarding paradigm (see Figure 6). Unicast addresses are used for one-to-one communications and currently carry the majority of IP traffic. Multicast addresses are used for one-to-many communications and in IPv6 a number of special multicast addresses play a significant role in basic network functions. Anycast addresses are a new type of address proposed in the IPv6 specification and are used for what is called "one-to-nearest" communications. Anycast addresses are used for communications in which a source sends traffic to only one of a set of possible destinations. In format, they are indistinguishable from unicast addresses; what makes them "anycast" is their assignment to multiple interfaces in the network topology.

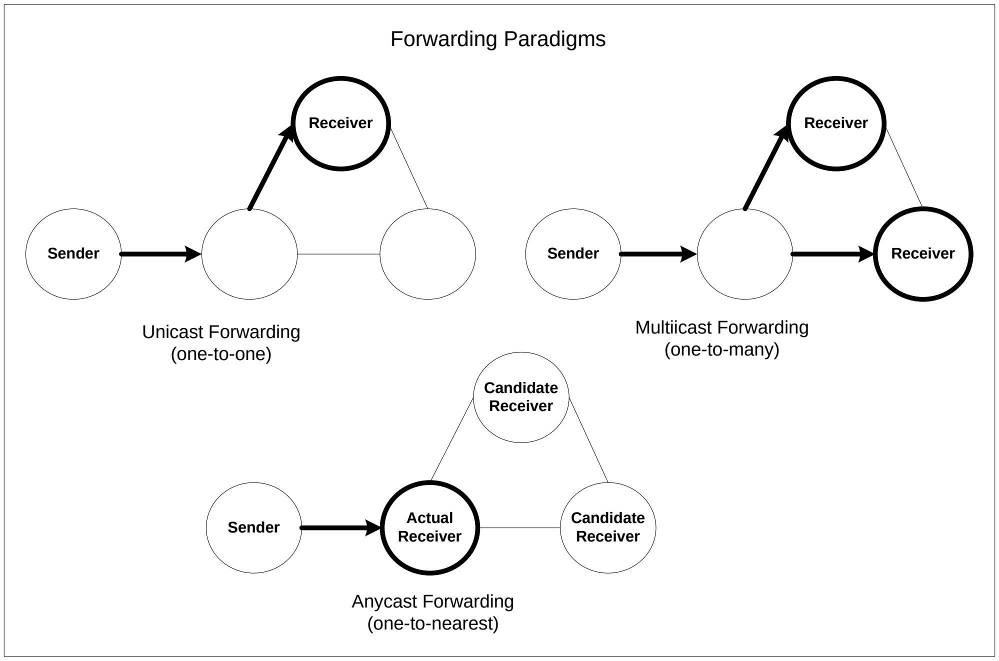


Figure 6: Forwarding Paradigms

4.3.5 Global Unicast Addressing Format

Global unicast addresses are intended to be globally reachable identifiers and communication endpoints for single interfaces. Their predefined format specifies a global routing prefix of 48 bits, a subnet identifier of 16 bits, and an interface identifier of 64 bits (see Figure 7) [Hinden 03]. As specified above, end-user organizations will each be assigned a global routing prefix ("/48"). These prefixes are aggregated into larger prefixes routed by ISPs. The subnet identifier allows the end-user organization to establish its own routing topology with over 64,000 segments. Each subnet has 64 bits of addressing capacity. The interface identifier may be assigned arbitrarily but the expected prevalent assignment mechanism is to derive the identifier from the interface's Layer-2 hardware address. The most common instance will be to map 48-bit IEEE 802 MAC addresses (such as those used in Ethernet) to the identifier, in what is referred to as "modified IEEE EUI-64" format. The ability in IPv6 to map hardware address to logical address is central to the stateless address auto-configuration (SLAAC) capability described below.

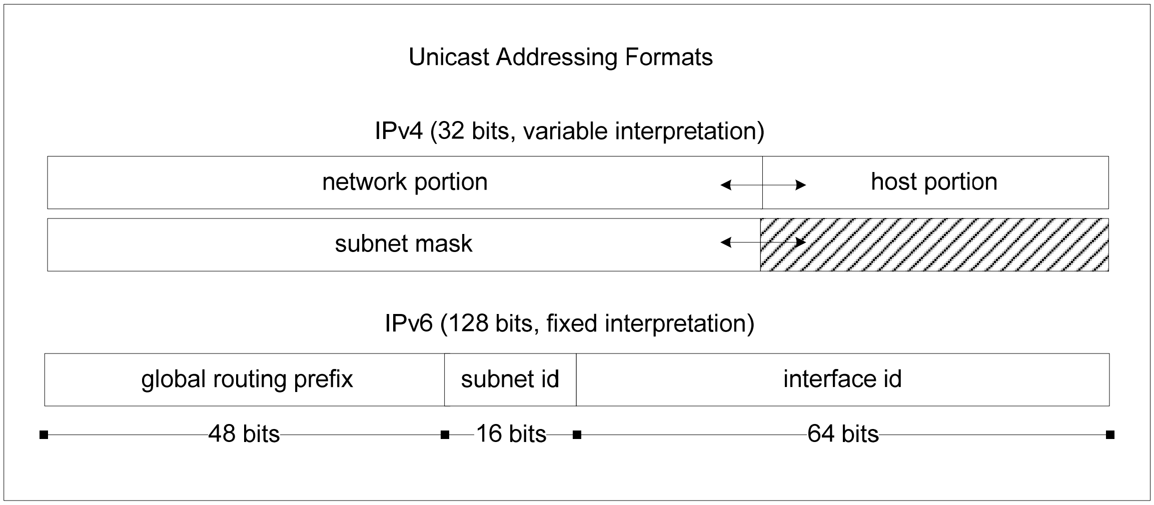

Unicast Addressing Formats
IPv4 (32 bits, variable interpretation)
network portion
host portion
subnet mask
IPv6 (128 bits, fixed interpretation)
global routing prefix
subnet id
interface id
48 bits
16 bits
64 bits

Figure 7: Unicast Addressing Formats

4.3.6 Scoped Addresses

IPv6 specifies two types of addresses whose scopes (or reachability) are less than global. The Unique Local Address (ULA) will be a globally unique but locally usable address [Hinden 05]. Its use is similar to "private" IPv4 addresses specified in RFC1918, except that because of the large space involved and randomization techniques there is a high level of assurance that the addresses will be unique per site. The ULA is useful for addressing interfaces for which it is not necessary or desirable to grant global connectivity. The uniqueness feature offers an improvement over IPv4 in that organizations that merge internal networks will not have to deal with renumbering. The link-local address, a new concept in IPv6, is an address mandatory for each interface which has a scope limited to the physical network link. It consists of a specially defined prefix plus a unique interface identifier, which is usually to be derived from the interface hardware address and can be generated by any IPv6 host without assistance from a router or a dynamic host configuration protocol (DHCP) server. The link-local address is the basis for key network functions related to the local network, most of which involve the critical neighbor discovery protocol (ND) described in more detail below [Narten 98]. These key functions include local router-to-host communications functions, local host-to-host communications functions, and auto-configuration of addresses.

4.3.7 Address-to-Interface Association

IPv6 introduces a new model for the association of network addresses with interfaces. First, as stated above, in comparison with IPv4 there will be a much closer association of the interface hardware address with the network address because of the interface identifier concept. Second, multiple simultaneous addresses per interface will be the “norm rather than the exception” in IPv6 [Davies 06b]. In addition to a unicast address, other address types will be associated with the interface, including two mandatory addresses: the link-local address and a special multicast address described below. There may also be multiple simultaneous unicast addresses associated with the interface, and these can be either global or ULA. One consequence of the many-to-one association of addresses to interface is the need for standard default address selection algorithms and mechanisms, which are specified by Draves [Draves 03]. Third, the address-to-interface association consists of two valid states (“preferred” and “deprecated”) and the duration of this association is governed by a two-phase lifetime, where the phases correspond respectively to the two valid states. The purpose of the two valid states is to assist in sustaining connectivity during common transitional operations such as renumbering [Chown 06]. The states are also factors in the address selection mechanism. Finally, an option has been developed for “privacy addresses” which make use of these lifetimes to change the interface identifier frequently enough to avoid usage tracking [Narten 01].

4.3.8 IPv6 Specifications Relevant to Addressing

A number of IPv6 specifications are mechanisms which are involved in fundamental IPv6 addressing operations. SLAAC is a method which allows IPv6-capable hosts to self-configure their own addresses and other configuration information and thus reduce administrator workload [Thomson 98]. In the SLAAC process, a host connected to a network first generates its link-local address, which allows it to communicate with other hosts local to the link, including a router. The router, through the use of a Router Advertisement (RA) informs the host of what information should be auto-configured, including whether the address should be obtained in a stateless or stateful (DHCP server) manner. In the stateless case, the RA also provides information on which prefix is associated with that local subnet, as well as lifetime information. The host combines this prefix information with its self-generated interface identifier to create a unicast address (see Figure 8). SLAAC is largely made possible through the use of the RA, which is a part of the Neighbor Discovery protocol (ND) [Narten 98]. ND is a formal set of functions which perform various operations among hosts and routers on a local link. A subset of these operations is accomplished in IPv4 through a set of disparate mechanisms including the rarely used Internet Control Message

Protocol (ICMP) Router Discovery Protocol (IRDP) and address resolution protocol (ARP). Key functions include

- router discovery (discovery of link information from routers, including information for SLAAC)
- address resolution (resolving link-layer addresses from network-layer addresses)
- next-hop determination (deciding whether or not a datagram should be delivered to the router)
- neighbor unreachability detection (monitoring reachability of other nodes on the local link)
- duplicate address detection

The latter two functions are new to IPv6. The address resolution function is accomplished through a completely new method. Instead of using an ARP broadcast (which does not exist in IPv6), a host sends a Neighbor Solicitation (NS) containing a neighbor's mandatory solicited-node multicast address to obtain its Layer-2 hardware address sent in a Neighbor Advertisement (NA). The solicited-node multicast address is generated from a well-known prefix and the lower 24 bits from a host's interface identifier. The next-hop determination function in IPv6 uses prefix information from RA messages instead of the subnet mask (which is used in IPv4). The unreachability detection function involves a neighbor cache which retains information on the reachability of local link neighbors and is more frequently updated than IPv4's corresponding ARP cache. Duplicate address detection is a check performed during the SLAAC process and uses NS and NA messages to verify local uniqueness. Finally, it should be noted that ND is primarily implemented through the use of ICMPv6 messages [Conta 06]. Thus ICMPv6 has a much more critical role than ICMP in local network functions, including functions related to network addressing.

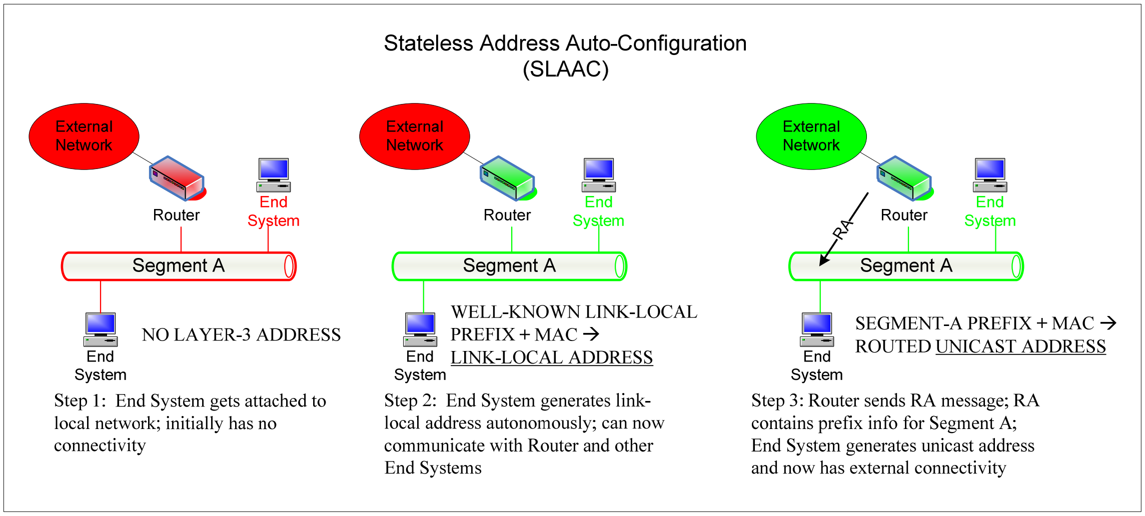

Stateless Address Auto-Configuration (SLAAC)
External Network
Router
End System
Segment A
End System
NO LAYER-3 ADDRESS
Step 1: End System gets attached to local network; initially has no connectivity
External Network
Router
End System
Segment A
End System
WELL-KNOWN LINK-LOCAL PREFIX + MAC → LINK-LOCAL ADDRESS
Step 2: End System generates link-local address autonomously; can now communicate with Router and other End Systems
External Network
RA
Router
End System
Segment A
End System
SEGMENT-A PREFIX + MAC → ROUTED UNICAST ADDRESS
Step 3: Router sends RA message; RA contains prefix info for Segment A; End System generates unicast address and now has external connectivity

Figure 8: Stateless Address Auto-Configuration (SLAAC)

4.4 Architectural Properties of Networks Using IPv6 Addressing

From the technology features described above, we can identify five general architectural properties of IP networks to which IPv6 makes fundamental changes. These architectural properties and IPv6's effects on them will form the framework for the subsequent operational survivability analysis.

4.4.1 Increase in Size of Address Space

As mentioned above, the 128-bit size of the IPv6 address is not solely a vast expansion of available addressing capacity. Aside from improving the sheer scalability of addressing network interfaces, the expanded space is an opportunity for increased structural flexibility; it brings the ability to assign purposes to various bit ranges without the constraint of address space conservation concerns. From the scalability and flexibility made possible by the 128-bit size proceed all of the other properties listed below. Finally, as described above, the 128-bit size has resulted in a more unwieldy human-readable representation, resulting in potentially degraded manageability.

4.4.2 Hierarchical Routing and Addressing

The structural flexibility of IPv6 address space allows for the creation of a strict hierarchy in both addressing and network topology, in which top-level ISPs aggregate and exclusively route prefixes for downstream entities, including other ISPs and end-user organizations. In the IPv4 Internet, addressing prefixes are generally not indicative of network topology. Although blocks of addresses are assigned to ISPs, a great number of these blocks have been fragmented into smaller pieces routed by other ISPs for a number of operational reasons, including difficulties in renumbering and multi-homing. The result is that while some address prefix may be routed out of one ISP, a more specific prefix within that block may be routed out of another. Consequences include a lack of correspondence between addresses and network topology, global routing tables containing hundreds of thousands of prefixes, and ultimately significant operational complexity. Since the Internet Registries are allocating IPv6 addresses according to strict guidelines and addresses are not considered portable, an IPv6 allocation will be represented as a prefix uniquely routed by an ISP. Thus addressing prefixes assigned to end-user organizations ("/48's") might be easily aggregated into single, larger prefixes, resulting in a hierarchical routing system in which addressing mirrors the topology (see Figure 9).

In such a system, there are also far fewer routing prefixes to manage and maintain in global routing tables. Both of these properties, alignment of addressing and topology and scalability of routing, allow for potentially enhanced manageability. Within an end-user organization, the number of available subnets in a /48 allocation (16 bits or over 64,000), each of virtually unlimited size (64 bits), also allows that organization to develop internal addressing allocation policies similar to this scheme. It is important to note that counter to the hierarchical routing model, the IPv6 anycast specification by its very nature undoes the strict alignment of addressing and topology, since the assignment of anycast addresses removes the uniqueness property of unicast addresses in the topology.

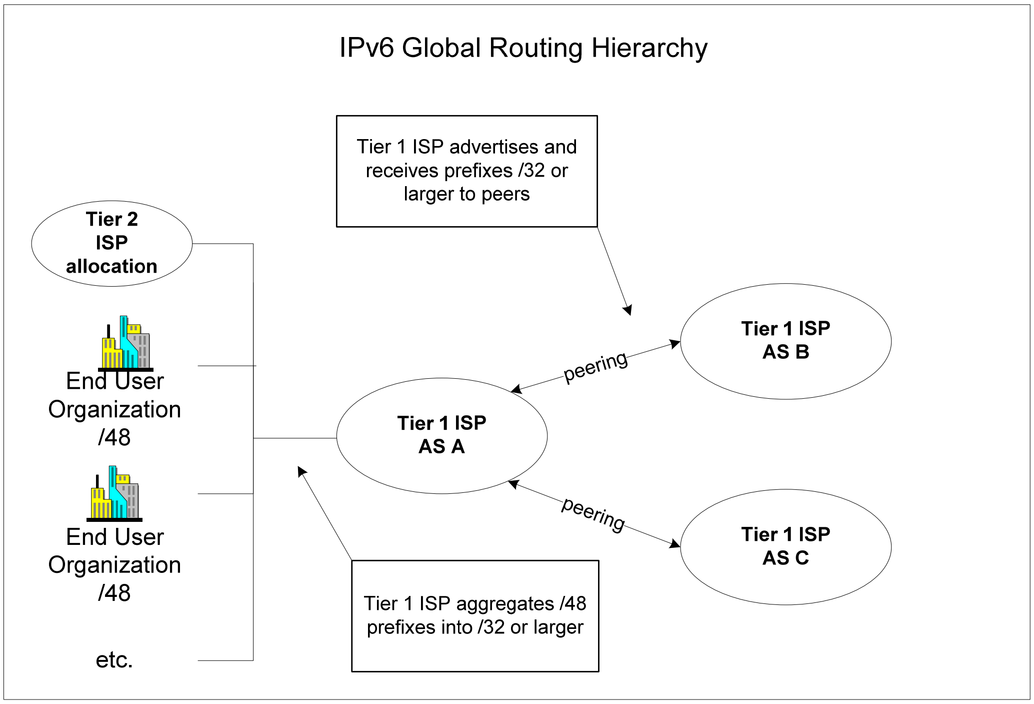

IPv6 Global Routing Hierarchy
Tier 1 ISP advertises and receives prefixes /32 or larger to peers
Tier 2 ISP allocation
End User Organization /48
End User Organization /48
etc.
Tier 1 ISP AS A
Tier 1 ISP AS B
Tier 1 ISP AS C
peering
peering
Tier 1 ISP aggregates /48 prefixes into /32 or larger

Figure 9: IPv6 Global Routing Hierarchy

4.4.3 Increased End-to-End Transparency

The virtually inexhaustible space afforded by IPv6 addressing generally obviates the need for address conservation technologies such as NAT and its variants [Egevang 94]. The use of NAT in the IPv4 Internet (as well as other technologies, such as stateful firewalls and proxies) has resulted in a mostly non-transparent network architecture, in which intermediate systems handle and often modify a significant amount of connection state between end systems [Hain 00]. In a network which is end-to-end transparent, a host and its resident network applications assume that a unicast address is a unique and durable communication endpoint identifier, and that traffic destined to that address would for the most part reach that endpoint unaltered [Carpenter 00]. The use of global unicast addressing in IPv6 presents the possibility of restoring a measure of end-to-end transparency by increasing the general uniqueness and durability (or persistence) of addresses. The increased presence of end-to-end transparency in turn may influence other architectural properties, including operational transparency (which goes hand in hand with the cognizance element of operational control), overall robustness (due to the reduction of connection state on intermediate nodes) and end-to-end security (the ultimate goal of network-layer security as implemented in IPsec) [Van de Velde 05], which in turn may influence the prevailing perimeter model of network security. The architectural and operational effects of the non-transparency resulting from the wide deployment of NAT have been well documented in Hain and will be discussed in more detail below [Hain 00]. Note also that proposals such as “privacy addresses” will act against the end-to-end transparency property [Narten 01].

4.4.4 Increased Transparency, Autonomy, and Complexity of End Systems

The changes in the association of IPv6 addresses to interfaces described above have multiple architectural effects on the network as a whole. The closer association between the link-layer hardware address and the network-layer logical addresses assigned to the interface, brought about by the interface identifier concept, potentially increases operational transparency, since a network-layer address based on interface identifier is not an arbitrary assignment as it is in IPv4. In addition, the mechanisms exploiting the closer association, including link-local address generation and SLAAC, enable increased autonomy for end systems, in both interface configuration and link-local network communications. The many-to-one relationship of logical addresses to the end system interface, due to multiple addressing types as well as the availability of multiple instances of one addressing type per interface, increases the complexity of the end system, realized in the increased complexity of both the internal protocol stack and the external management model. The likelihood of the long-term co-existence of IPv4 and IPv6 on networks suggests that many end systems (particularly servers) will either have “dual stacks” or virtual tunneling interfaces, which further increases end system complexity [Bound 06]. Finally, the existence of multiple states and attendant lifetimes in the address-to-interface association also increases the complexity of the end system. Because these mechanisms influence the persistence attribute of the address-to-interface association, both operational and end-to-end transparency may be affected depending on how they are employed.

4.4.5 Increased Automation in the Local Network

Closely related to the increased autonomy of end systems is IPv6’s formalization and integration of the ND and ICMPv6 protocols into operation of local network functions. These include modified functions under IPv4 (router discovery, next-hop determination, and local address resolution) and new functions under IPv6 (neighbor unreachability detection and duplicate address detection). ND directly manages these functions and uses ICMPv6 as its message-passing mechanism. The intention of these protocols is increased automation of local network functions. A concomitant effect is the network’s increased dependency on protocol message integrity and availability. One IETF Internet Draft in fact describes ICMPv6 as “essential to the functioning of IPv6 rather than a useful auxiliary” [Davies 06b]. This dependency has obvious operational implications, which will be discussed below.

4.5 Survivability Analysis

It should be apparent from the discussion in Section 4.4 that the architectural properties of IPv6 will have significant effects on the operational control of IP networks. In the following analysis, system survivability will generally be evaluated in terms of the constituents of operational control discussed in Section 4.2 (cognizance, analysis, and execution capability), although critical issues affecting the architectural survivability of IP networks will be discussed where necessary. The analysis comprises five sections, one each for the five broad architectural categories described in Section 4.4.

Before proceeding with the analysis, it should be noted that there will be significant operational (and therefore survivability) effects not arising from IPv6 addressing architecture per se, but rather from the novelty of the technology as well as the inherent complexity of transitional environments. These include the general lack of operational experience among support staff and the lack of mature IPv6 software supporting network and system management functions, and in particular the lack of essential features from early versions of this software (for a good example see [Davies 06b]). The co-existence of IPv4 and IPv6 on transitional networks will generally involve some combination of dual-stack nodes and tunneling, analyzed by Bound and associates [Bound 06]. It is obviously critical that sufficient management and technical support infrastructure be in place before committing a network to this transitional environment.

4.5.1 Increase in Size of Address Space

Operational factors related to 128-bit addressing space size include the increase in scale for addressing network interfaces and the change in the human-readable representation of an address. In addition, the expanded size results in derivative architectural properties which in turn have other operational effects.

The considerable increase in scale changes the characteristics for network management functions such as scanning and network admission control. The default 64-bit address space for each network segment will make current methods for host, port and service scanning computationally infeasible. Chown illustrates the difference using a single probe per second scenario, in which a typical 8-bit IPv4 subnet can be scanned in about five minutes, and a 64-bit IPv6 subnet will take roughly five billion years to scan. The same source also notes that IPv6 networks may be inherently more defensible against worms which use scanning techniques to acquire targets [Chown 05]. Thus the larger address space potentially reduces the cognizance of both attackers and legitimate auditors of subnets. The infeasibility of scanning subnets remotely is countered by the facility for scanning from a node present on the subnet itself, by using the Neighbor Discovery protocol’s (NDP) Neighbor Solicitation (NS) message to the “all-hosts” multicast address. For the legitimate auditor, the requirement to perform scans on-link is a net reduction in execution capability which may be partially remedied through a distributed architecture for controlling local scanning hosts. The “all-hosts” multicast address may of course be exploited by either an insider or an intruder into the Layer-2 network. Once a host on a subnet is compromised, the size of the address space becomes a non-factor for the ability of the attacker to scan. Furthermore, it is possible for an attacker to reduce the search space for a remote scan, either through guessing that the administrator has assigned a predictable range of addresses or, in the case of SLAAC, making educated guesses about the hardware address vendor code (which will appear in the interface identifier), using DNS information found in public servers, and harvesting addresses from sniffed traffic or system logs [Chown 05].

The 64-bit space for each network segment will also reduce the efficacy of common admission control mechanisms such as ingress filtering and reverse path forwarding (RPF) checks. Both of these mechanisms are general purpose, broadly targeted means for controlling the source addresses of non-local traffic from each network segment. Default implementations of both mechanisms allow admission to the full range of on-link addresses, so that attackers have the entire 64 bits of interface identifier space to use for source address spoofing. This reduction in execution capability could be addressed by other mechanisms, such as dynamic access control, where a router’s ingress filter becomes a “default-deny” access control list configured on demand by explicit updates from a registration server. Essential to this method of admission control is the deployment of strong Layer-2 access control and authentication mechanisms such as 802.1X. The need for link-layer control is particularly critical in wireless environments or environments with insufficient physical security.

The size of 128-bit addresses may also present difficulties for protocols which embed IP addresses. For instance, Simple Network Management Protocol management information bases (SNMP MIBs) often use IP addresses as indices, and it may be more difficult to design management objects because of size constraints to index fields [Roese 04]. Inasmuch as management protocols such as SNMP are restricted, both cognizance and execution capability may be reduced.

The relatively unwieldy representation of 128-bit addresses may result in degraded manageability for human-conducted functions in network management. IPv4 four-octet dotted-decimal representation partially facilitates memorization or at least recognition of addresses. The use of eight-word hexadecimal representation, even with compression techniques, makes addresses far less recognizable and subject to more human error in reading and writing configurations. In this respect, cognizance and execution capability are degraded. On the other hand, since the structure of IPv6 addresses delimits the global routing prefix, subnet identifier, and interface identifier on pre-defined word-boundaries, the human analyst is able to extract these identifiers more easily than classless IPv4 addresses, which require familiarity with applying subnet masks in order to determine the network and host portions of the address. The effect is enhanced analysis of network addresses and configurations which use them. Nevertheless, IPv6 address representation may complicate management of address-centric configurations (such as firewalls or intrusion detection systems). Even with IPv4 address representation, visual analysis of long filter sets is a painstaking task; with IPv6 representation, human analysis may become impractical. Therefore other methods for managing address-centric configurations are necessary. Configurations with significant numbers of IPv6 addresses will likely have to be managed using abstractions which aggregate network addresses into manageable objects.

The structural flexibility which arises from IPv6's expanded addressing capacity adds functionality to the network which enhances operational control. The functionality is a result of the ability to define special ranges of address space with well-known prefixes. In addition to the link-local and solicited-node multicast addresses, which are discussed in more detail below, IPv6 provides the unique local address (ULA), which is an enhanced version of IPv4 private addressing. Since generated ULAs have a high assurance of being globally unique, organizations using them may interconnect and not have to deal with addressing conflicts and attendant operational difficulties. The use of a well-known prefix for the ULA also makes filtering the prefixes simple. The ULA thus enhances execution capability for merging internal networks, one of the most difficult network operations. In addition, the ULA can be used to add architectural survivability for site-local connectivity in the face of lost or intermittent global connectivity [Van de Velde 05]. Finally, as mentioned in Section 4.3.1, the abundance of IPv6 addressing capacity removes the main motivation behind the deployment of NAT. A widespread reduction in the use of NAT would lead to greater end-to-end transparency across networks. The operational effects of this architectural property are discussed below.

4.5.2 Hierarchical Routing and Addressing

The alignment of addressing and topology and the improved scalability of routing promote enhanced operational control both for the global Internet and inside individual site networks. Due to the need for address conservation, in IPv4 multiple and disparate logical subnets are often overlaid in a single physical network segment. Unless these subnets are bit-aligned and therefore aggregable, upstream routers will contain multiple prefixes per network segment. In IPv6 there is a single prefix per network segment. The one-to-one association of prefixes and network segments, as well as the minimization routing table size enhance both cognizance and analysis of a network's routing state. This improvement provides the greatest benefit to Tier-1 ISPs responsible for managing global routing tables. Since allocations for end-user organizations are set as /48 size blocks, and allocations for ISPs as /32 size blocks, a Tier-1 ISP's internal routing tables should contain one /48 prefix per downstream site, and its global routing tables ideally contain a limited number of /32 prefixes from each of its peers. In contrast, a Tier-1 ISP in the current IPv4 must manage hundreds of thousands of routes in its global and internal routing tables. The improvements to cognizance and analysis brought about by simplifying global routing should improve execution capability for critical functions such as back-tracing denial-of-service attacks and configuring peering policies [Van de Velde 05]. Architecturally, the presence of fewer and purely aggregated prefixes advertised at peering points should result in greater stability for the routing system, since there is a much smaller probability that route flipping within one autonomous system (AS) would be propagated to peer AS's. It should be noted that the IPv6 anycast specification works against the notion of aligning addressing and topology, since by definition an anycast address is a unicast address assigned to more than one interface. A source sends traffic to the "nearest" (in terms of routing metrics) interface configured with an anycast address, and it is expected that within a routing domain anycast addresses will be represented on routing tables by "host routes" (i.e., single addresses) [Hinden 06]. Since the use of anycast addresses adds multiple prefixes not associated with a specific network link, it increases the size and complexity of routing tables. Troubleshooting connectivity to anycast addresses may also be difficult because of the indeterministic nature of the service. In these respects, an anycast deployment has the potential to reduce all aspects of operational control. Planners should carefully weigh the benefits of an anycast deployment against the operational difficulty of maintaining the service.

4.5.3 Increased End-to-End Transparency

End-to-end transparency is an architectural property which conforms to the end-to-end network design principle, which states that "certain functions can only be performed in the endpoints; thus they are in control of the communication, and the network should be a simple datagram service that moves bits between these points" [Hain 00]. In other words, the endpoints should exclusively maintain connection state. This design principle is undone by NAT (as well as other intermediate devices such as stateful firewalls), and variants such as NAPT (network address port translation), which maintain connection state in the interior of the network (see Figure 10). End-to-end transparency is a property which enables the end-to-end model and is characterized by both the uniqueness and durability of endpoint network addresses. Overloaded NAT and NAPT take away both properties. Restoring address uniqueness and durability with the global unicast model of IPv6 will therefore restore the operational transparency (that is, cognizance and execution capability) brought about by endpoint transparency. For instance, endpoint transparency will make network traffic flows easily traceable [Van de Velde 05]; through a NAT, tracing traffic flows requires the analyst either to have access to non-persistent translation state in real-time or to have access to translation logs from the NAT node and make temporal correlations [Van de Velde 05].

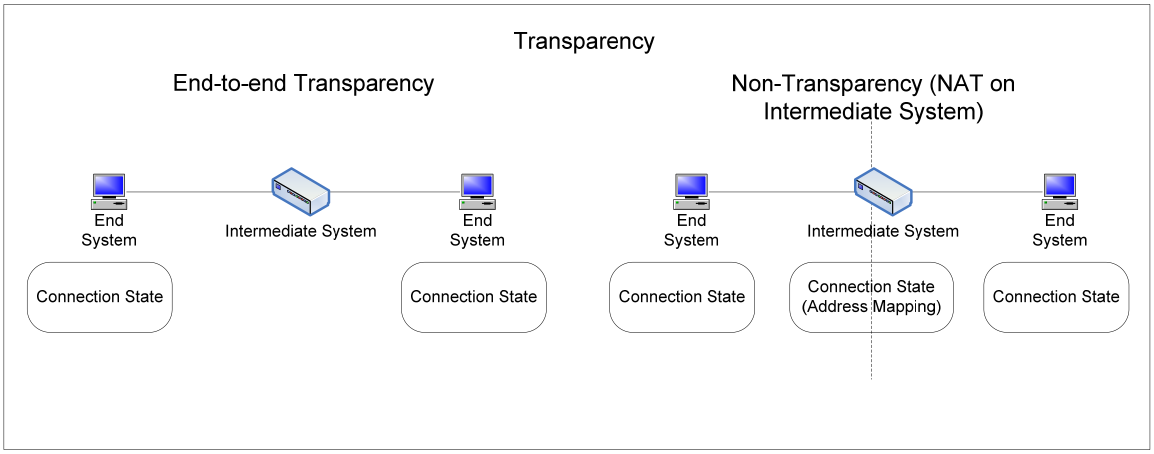


Figure 10: End-to-End Transparency as Undone by NAT

In addition to reducing the management complexity inherent in NAT, end-to-end transparency also encourages a network security model which promotes cognizance of explicit state over reliance on implicit state. Hain argues against the “perceived security...from the lack of pre-established or permanent mapping state” in favor of user-friendly stateful firewalls at endpoints and properly configured intrusion detection and content-aware filtering in the network: “to the degree that [these] techniques improve a network manager's ability to explicitly know about or control access, and thereby manage the overall attack exposure of local resources, they act to improve local network security [Hain 00]. In particular the explicit nature of a content-aware firewall...will be a vast security improvement over the NAT artifact where lack of translation state has been widely sold as a form of protection” [Van de Velde 05].

The removal of NAT from the network also improves architectural survivability. In terms of the end-to-end network design principle, NAT becomes a critical single point of failure because it maintains the state of multiple connections which do not recover as they would in a network where intermediate nodes simply forward traffic and are able to route around failures [Hain 00]. Network scalability is also improved by removal of NAT, since network growth leads to growth of the connection state which NAT devices in the core of the network need to maintain, making them not only single points of failure but also performance bottlenecks [Hain 00]. Furthermore, the criticality of NAT devices makes them and their address pools valuable targets of attack [Van de Velde 05].

Corollary to end-to-end transparency is the possibility of another architectural property with implications for operational control: end-to-end security (see Figure 11). With end-to-end transparency there is an opportunity to provide host-to-host confidentiality and authenticity services at the network layer using IPsec. IPsec is a required component for the IPv6 protocol stack and is essentially incompatible with NAT, which modifies packet headers [Davies 06b]; also, IPv4 deployments of end-to-end IPsec often require the use of application-specific workarounds such as NAT Traversal. An environment favorable to end-to-end security would free developers to build more advanced peer-to-peer applications which require security [Van de Velde 05]. However, it should be noted that the key distribution problem remains a major barrier to widespread deployment of end-to-end security.

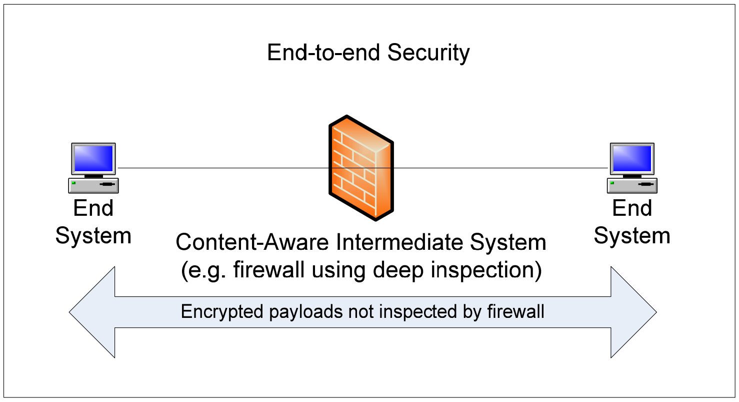

End-to-end Security
End System
Content-Aware Intermediate System (e.g. firewall using deep inspection)
End System
Encrypted payloads not inspected by firewall

Figure 11: End-to-End Security

The greatest potential impact of widespread end-to-end security is its effect on the prevailing perimeter model of network security. If a substantial amount of network traffic uses end-to-end security, intermediate network security devices such as firewalls and intrusion detection systems will be rendered ineffective because they can no longer perform deep packet inspection. Davies suggests that in general it is unsafe to rely on the perimeter model for security, and states that developments in technologies such as "trusted computing" and security policy distribution signal a shift to reliance on host-based security mechanisms including host-based filtering and intrusion detection [Davies 06b]. The fundamental shift of the perimeter model to the host-based model means that the network security roles of perimeter and end systems will change. While the shift will not automatically result in net positive or negative effects on operational control, it changes the management model for the objects of cognizance, analysis, and execution capability. In addition to direct, centralized control of perimeter and core nodes, the new model will require the indirect, distributed control of agents at end nodes. This model is analogous to the familiar C2 tenet of centralized control and decentralized execution [USAF 01]. In order to maintain operational control under a host-based model, reliable and secure distributed control mechanisms on intermediate systems, as well as user-friendly security mechanisms on end systems, must be in place.

As we have seen, an IPv6 environment will remove the address conservation motive for using NAT and end-to-end transparency promises considerable benefits for operational and architectural survivability. However, it should be noted that there are proponents of NAT who consider it a security technology. It is recommended that the tradeoffs between NAT and end-to-end transparency be carefully considered if a NAT-based security model is being cited as a reason for not migrating to IPv6. Informative sources endorsing the end-to-end transparency perspective include Van de Velde and Hain [Van de Velde 05, Hain 00].

4.5.4 Increased Transparency, Autonomy, and Complexity of End Systems

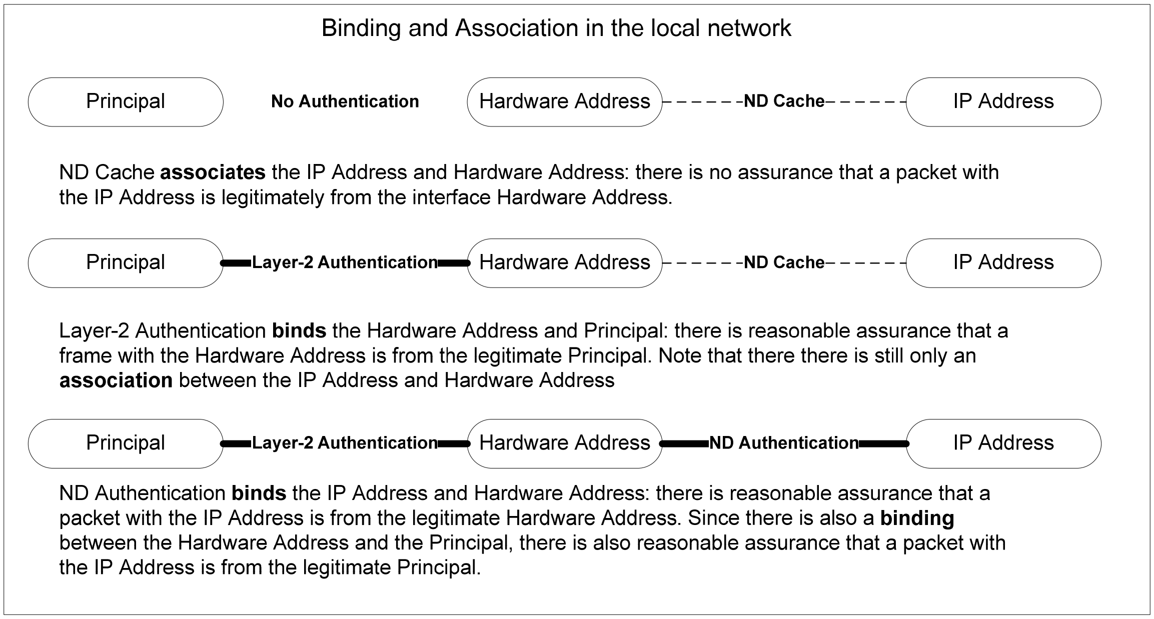
The analysis of IPv6 addressing and end-to-end transparency posits that security functions will become more prevalent on end systems. In a parallel fashion, under IPv6 addressing architecture, end systems will assume a greater role in local network functions. The expansion of the end system's role arises from the opportunity in IPv6 to associate the link-layer hardware address and various network-layer logical addresses by means of the interface identifier. As a result the end system is more transparent, enhancing cognizance and execution capability for operations using network addresses. For instance, a common operation is to trace IP traffic to a physical switch port. In an IPv4 environment, the IP address must be associated with a link-layer hardware address in an ARP table, and then the link-layer address must be associated with a switch port in a MAC-to-port table. Since the interface identifier is derived from the link-layer hardware address, one fewer association has to be made for a trace. It is important to note that the interface identifier is an association and not a binding. In order to bind the identifier to the interface it claims to belong to, authentication mechanisms are required. It should also be noted that that the privacy extensions of SLAAC, which allow for the generation of pseudorandom interface identifiers at periodic intervals, degrades the durability of the association and therefore negates the cognizance and execution capability afforded by interface identifiers (e.g., maintenance of access control lists) [Narten 01]. In an environment which relies upon the association implied in the interface identifier, privacy extensions should not be used.

In combination with well-known prefixes, the interface identifier is also used in IPv6 mechanisms which increase the autonomy of the end system. The link-local address can be independently generated by a host interface and therefore the host can initiate local network-layer communications without any external control or awareness, and without any dependency on RA messages for on-link information [Davies 06b]. Clearly the ability of hosts to communicate autonomously and automatically on the local link reduces cognizance of local network activity. If the local link is wired, it is possible to reassert some measure of control by having hosts use link-local addresses solely for local network management functions [Davies 06b]. Interface configuration through SLAAC is also a function which makes use of link-local addresses. Because SLAAC allows hosts to configure global unicast addresses autonomously, the traffic scope of the self-configured host is no longer restricted to the local link, and a malicious node so configured accordingly has a much wider scope to cause damage. Without other external monitoring and control mechanisms and policies, administrators of a SLAAC-enabled network stand to lose a considerable amount of cognizance and execution capability in host configuration functions. Administrators may need to compensate for this loss of visibility by performing regular audits of network segments (e.g., executing ICMPv6 echo requests to the “all-hosts” multicast address). Furthermore, SLAAC may diminish operational control in other network functions. For instance, securing dynamic DNS updates from hosts configured with SLAAC will be difficult to achieve [Davies 06b]. On the other hand, the main purpose of SLAAC—to save scarce administrator time—may also enhance overall operational control by freeing administrators to manage other critical tasks. It should also be noted that host configuration does not necessarily have to be accomplished purely through SLAAC; it is possible to use a combination of stateful and stateless approaches which saves administrator time but maintains administrator cognizance and execution capability in host configuration [Thomson 98]. As mentioned previously, the use of privacy extensions in SLAAC per Narten will reduce operational control, and the only method to prevent their use is through a full stateful approach using DHCPv6 [Narten 01, Davies 06b]. Finally, it should be noted that giving configuration autonomy to the end system implies trust. Therefore, if a network decides to deploy SLAAC, it is crucial to implement link-layer authentication and access control mechanisms such as IEEE 802.1X, especially in wireless environments [Davies 06b]. In summary, SLAAC may entail a significant loss of operational control, and deciding whether and how to deploy SLAAC or not warrants a careful consideration of the tradeoffs listed above.

The autonomy of end systems under IPv6 is accompanied by corresponding complexity. Both the many-to-one association of logical addresses to the end system interface and the multiple states and lifetimes of these associations make possible a number of functions which extend execution capability for a wide variety of operations such as renumbering, multi-homing, and multicast distribution of policy [Roese 04]. But the added functionality comes at the price of greater complexity, which comes in two forms. First, the IPv6 protocol stack requires standard address selection mechanisms to deal with multiple addresses, address types, and phases per interface [Draves 03]. In addition, there must be a policy management interface and applications to override default address selection, which potentially increases operational complexity [Roese 04]. Even with such software, the presence of multiple addresses of the same type on a single interface will likely degrade cognizance and obfuscate analysis of host behavior. Second, multiple addresses per interface complicates external management. In a many-to-one association, the address may be unique to the interface, but the interface is not unique to the address, which means that policy configurations using network addresses as unique host identifiers must take into account all of the addresses associated with a host interface. Host interfaces with multiple addresses have the potential to degrade all three elements of operational control, since there is no single identifier which can be used in mechanisms to enforce uniform policy (such as access control lists) for the collection of addresses resident on the host interface [Davies 06b]. Therefore configurations which use IP addresses as implicit host identifiers should combine multiple addresses into “policy objects” to simplify management and retain operational control.

4.5.5 Increased Automation in the Local Network

IPv6 and its “helper” protocols ND and ICMPv6 are designed to facilitate automation in the local network. The previous analysis of SLAAC alludes to the benefits of automation in freeing scarce administrator time and thus improving overall operational control. In addition to host configuration, two functions implemented in ND, neighbor unreachability detection and duplicate address detection, are unavailable in IPv4 and are handled by manual procedures. Their automation in IPv6 is a net gain for operational control, since it frees administrators from common and time-consuming tasks and at the same time does not sacrifice cognizance of local network state. The most significant operational effect of increased automation on the local network has to do with the network’s increased dependency on the integrity and availability of ND/ICMPv6 messages which perform critical local functions. Threats involving malicious ND/ICMPv6 messages (e.g., attacks similar to ARP cache poisoning) may disrupt operational control of local networks. See Nikander for a lengthy description of these threats [Nikander 04]. A solution called Securing Neighbor Discovery (SEND) has been proposed to authenticate ND/ICMPv6 messages, but it is only applicable for binding the messages to Layer-2-authenticated nodes [Arkko 05]. To provide true authentication for ND/ICMPv6 messages, the nodes themselves must also be authenticated (the Layer-2 hardware addresses must be bound to a legitimate user) when they connect to the local network (see Figure 12). Again, the employment of link-layer authentication and access control mechanisms will be critical in IPv6-addressed network segments, particularly in unsecured physical network environments.



ND Cache **associates** the IP Address and Hardware Address: there is no assurance that a packet with the IP Address is legitimately from the interface Hardware Address.

Layer-2 Authentication **binds** the Hardware Address and Principal: there is reasonable assurance that a frame with the Hardware Address is from the legitimate Principal. Note that there there is still only an **association** between the IP Address and Hardware Address

ND Authentication **binds** the IP Address and Hardware Address: there is reasonable assurance that a packet with the IP Address is from the legitimate Hardware Address. Since there is also a **binding** between the Hardware Address and the Principal, there is also reasonable assurance that a packet with the IP Address is from the legitimate Principal.

Figure 12: Binding and Association in the Local Network

4.6 Conclusions

4.6.1 General Observations

As argued in the Introduction, the survivability of networks underlying JBMC2 relies particularly on system attributes which are operational in focus, because they will be complex systems of systems operating in dynamic environments characterized by uncertainty. The inherent properties of complex systems of systems—unintended consequences, architectural mismatches and semantic incompatibility—require human agents and not automated processes to resolve the inevitable conflicts that arise in this environment. Thus the role of human agents will be critical for the foreseeable future, even when technologies such as IPv6 offer mechanisms to automate network management. If human agents are the critical elements of these systems, it makes sense to build technical mechanisms which support rather than replace them.

In other words, network management systems should be designed more for improving the cognizance, analysis, and execution capability of human agents rather than for functional comprehensiveness (“feature creep”) or complex integration with other systems. Analysts do not need more knobs and buttons, nor do they need knobs and buttons that automate or integrate multiple tasks; rather, they need simple knobs and buttons that will do exactly what they expect them to do. Therefore automation of most network management tasks in complex systems-of-systems environments is desirable insofar as the mechanism saves administrators from simple, repetitive tasks, has predictable results, and does not increase the indeterminism of network state. At the current state of the art, automation is a tool, not a replacement for human agents. This is particularly true for technologies such as IPv6, for which there is relatively little collective operational experience.

At the same time, there are limits to human cognitive capacity, and as technologies primarily designed to address the increasing scale and complexity of networks (such as IPv6) are deployed, operational control will suffer unless more advanced models for controlling these technologies are adopted. As shown in Section 4.5, architectural changes brought about by IPv6 addressing architecture will likely increase the necessity of using abstractions such as management objects and deploying decentralized agents on end systems.

It should also be noted that the operational properties of IPv6 addressing architecture help and hinder both administrators and intruders, similar to the way terrain affects all combatants in land warfare. It is not possible to give a simple answer on the net effect of IPv6 addressing on operational control. Instead, the architectural properties and their survivability effects should be evaluated individually and in the context of an actual deployment.

Finally, this study generally deals with the architectural properties of a full native IPv6 environment. For all practical purposes, it is likely that IPv4 and IPv6 will co-exist in a semi-permanent state on the Internet. Although IPv4-to-IPv6 transition strategies are out of scope here, it is in this context that network planners should make conclusions about the recommendations herein.

4.6.2 Recommendations for IPv6 Deployment

The following recommendations are drawn from the analyses in Section 5. The recommendations share an emphasis on the primacy of operational control in the survivability of an IPv6 deployment.

- Conduct a careful analysis of stateful and stateless IPv6 address configuration mechanisms and consider the tradeoffs of each. In particular, do not be tempted into a pure SLAAC deployment in order to remove administrators from the host configuration process. Recall that hybrid stateless-stateful solutions are possible and that it is preferable to build a solution which balances efficiency and control. A good practice is to pilot a network segment with the solution to get a better idea of the tradeoffs involved. Finally, privacy extensions to SLAAC are not recommended for networks in JBMC2, but note that only a fully stateful approach can prevent their use.

- IPv6 addressing should be deployed first on non-critical networks. Support systems and tools for IPv6 are currently not as mature or fully-featured as their IPv4 counterparts and therefore the same level of operational control cannot currently be maintained for IPv6 networks. Deployment on non-critical networks will allow time for the development of these ancillary systems as well as provide operational experience. At the same time, planners should avoid lower security standards for these early deployments, and in order to gain more meaningful operational experience, they should treat these networks as if they were critical.

- IPv6 should be deployed in a way which maximizes operational transparency for end systems. In other words, IPv6 addressing policies should avoid practices or mechanisms which degrade the uniqueness and durability of network addresses as well as their alignment with network topology. At the very least, planners should perform a comparison of risks and benefits for deploying these practices or mechanisms. In particular, planners should proceed with caution when considering privacy extensions to SLAAC, anycast addressing, and NAT. Finally, it is recommended that interface identifiers be generated using IEEE EUI-64 format, since it avoids the security problem of predictable address configurations and provides closer association between the interface and the network address.

- The end-to-end transparency property of IPv6 addressing, its potential to bring about end-to-end security, and the growing autonomy of end systems should force planners to rethink the perimeter security model. Over-reliance on perimeter systems such as NAT should be avoided and security models should incorporate substantial attention to the operational control of end systems. The security model for networks in JBMC2 should assume that there are malicious insiders in control of end systems.

- The security model for an IPv6-addressed network focuses on end systems and therefore must employ strong link-layer access control and authentication mechanisms which bind the end system to a legitimate user in real time. Furthermore, because local network control traffic is critical to IPv6 operation, ND/ICMPv6 messages must be authenticated in the JBMC2 environment.

- Object-oriented management features have been widely available on enterprise-grade equipment (e.g., firewalls) for several years. In an IPv6-addressed network it should be standard practice to make use of policy objects in place of direct management of IP addresses.

- Planners should take advantage of publications available from the IETF “v6ops” working group, which has produced valuable analysis and documentation of IPv6 operational issues and best practices [IEFT 06].

4.6.3 Future Work

This study analyzes the architectural properties of IPv6 addressing architecture and their implications on operational control, defined here as a composite of three elements: cognizance, analysis, and execution capability. The same approach may be useful in other aspects of the IPv6 specification, including extension headers, end-to-end fragmentation, flow labels and QoS, and IPsec.

5 Conclusion

Across these four studies, several common themes on the characteristics of complex systems and networks emerge, each of which suggests an overall approach which should be considered when developing the GIG and its subsystems. Our general conclusion is that in order to make the GIG a practicable system in the near-to-mid term, planners and developers need to pay close attention to defining and/or limiting its scope. In particular, given the state of technology, the GIG's goals of universal interoperability and greater automation may be unrealistic and for the latter, perhaps undesirable.

Universal interoperability may be unrealistic because of the difficulties of the architectural mismatches described in Jennifer Franks' study of unanticipated consequences. Another limitation is the indispensable role of domain experts in the development and ongoing maintenance of GIG component systems. Architects of GIG systems must not proceed strictly from theoretical models but need to involve user communities from the start, as emphasized in Dawn Day's study of system testing. The importance of domain experts and testing methods for systems of systems does not stop at the initial development stage. As discussed, GIG systems will undergo continual evolution. The adaptability required for GIG systems suggests that the actual users and domain experts will need to be heavily involved, if not primarily responsible, for the continuing development of GIG system technologies.

Finally, the critical role of system users directly applies to the issue of automation in the GIG. Richard Ciampa's discussion of autonomous agents and Chris Tsuboi's discussion of operational survivability in networks both deal with the problem of determining the balance of control between automated processes and human agents. Both studies conclude that technologies for use in the GIG, such as autonomous agents and network management systems, must be limited in both autonomy and scope of application, largely because their indeterminism and unreliability outweighs their benefits, given the current state of technology. Such technologies should be developed as tools that serve human agents rather than replacements for them. Human users should not only be the primary beneficiaries of the GIG; they also should remain the entities primarily responsible for delivering these benefits.

Appendix: Glossary for IPv6 Survivability Study

address resolution: A local network function performed by a host. The host derives the Layer-2 hardware addresses of a local neighbor from the neighbor's Layer-3 network address. In IPv4, address resolution is handled by the address resolution protocol (ARP) and in IPv6 the function is handled through neighbor solicitation (NS) and neighbor advertisement (NA) messages used by the ND protocol.

admission control: The set of mechanisms which control the entrance of traffic into a network. Admission control mechanisms are normally deployed at the network edge.

aggregation: When referring to routing, aggregation is the combining of network prefixes on aligned bit-boundaries in order to summarize smaller, more specific prefixes into a larger, less specific prefix (also known as a "supernet"). Aggregation of prefixes allows for a clean routing hierarchy under IPv6.

ARIN: American Registry for Internet Numbers. The regional Internet registry for North America. ARIN is a non-profit organization which, among other things, manages allocation policy for Internet numbers (such as IP addresses and AS numbers).

ARP: See address resolution.

autoconfiguration: A network function which allows devices attaching to a network to configure network interfaces without manual intervention. Autoconfiguration may involve setting IP addresses, default router information, and server information. In IPv6, two types of autoconfiguration are defined: stateful and stateless. Stateful autoconfiguration involves the use of a server which maintains state information on assignments of addresses to devices. Stateless autoconfiguration (SLAAC) does not require a server and allows a device to generate its own addresses by combining its interface identifier with prefix information offered by a router. Also see interface identifier.

Autonomous System (AS): On the Internet, an administrative domain whose constituent networks share a common external routing policy. AS's are implemented through the use of AS numbers in router configurations.

CIDR: Classless Inter-Domain Routing (RFC1519). On the current Internet, the standard way for making IPv4 address allocations and routing more efficient. CIDR associates subnet masks with network prefixes in order to make interpretation of the prefix size variable. This flexibility allows prefixes to be aggregated, thus saving space in routing tables. It also allows more efficient use of address space, since segments can be appropriately sized for the actual number of hosts. Under the “classful” Internet, prefixes and subnet addresses were assigned under inflexible octet boundaries, making routing and addressing inefficient. CIDR notation takes a network prefix and appends a “/” and the number of network bits.

connection state: The set of volatile information in a system’s memory which maintains real-time parameters for an end-to-end connection with another system. One key element of connection state is the IP address at the other end of the connection.

dotted decimal notation: The human-readable form for IPv4 addresses. The notation consists of four octets of decimal numbers (range 0-255) delimited by periods.

dual stack: The presence of both an IPv4 and IPv6 protocol stack on the same system. Network devices with dual stacks are one solution for the co-existence of IPv4 and IPv6. Also see tunneling interface.

dynamic addressing: A method for assigning IP addresses on the fly to hosts on a network. Under IPv4, dynamic addressing service is usually provided by a server running dynamic host configuration protocol (DHCP).

dynamic DNS: A method for domain names to be mapped to dynamic IP addresses. Specified in RFC2136.

end system: A networked host which is situated at the edge of the network. Normally, end systems are the major sources and sinks for network traffic.

end-user organization: In ARIN terminology, a node at the lowest level in the global routing and address allocation hierarchy. End-user organizations connect to the Internet and receive addresses through service providers.

hexadecimal notation: The human-readable form for IPv6 addresses. The notation consists of eight 16-bit words of hexadecimal numbers (range 0-FFFF) delimited by colons.

forwarding path: The sequence of Layer-3 hops taken by a datagram from its source to its destination(s). Usually denoted by a series of IP addresses belonging to intermediate systems (mostly routers).

global routing prefix: In the IPv6 specification, the first 48 bits of the 128-bit global unicast address. The global routing prefix represents all hosts in an end-user organization and is advertised to a service provider, which in turn may aggregate the prefix into larger prefixes.

hardware address: A globally unique number assigned to a network interface for purposes of Layer-2 communication. The most common type of hardware address is the IEEE 802 MAC address, which is 48 bits in length and used in technologies such as Ethernet. The first 24 bits of an IEEE 802 MAC address are a vendor code and the last 24 bits are a serial number assigned by that vendor. Hardware addresses may be spoofed in Layer-2 frames.

ICMP: Internet Control Message Protocol. A “helper” protocol for IP which provides network management functions, most of which involve error messages. ICMPv6 is the modified version of the protocol for IPv6.

IEEE 802 MAC address: See hardware address.

IEEE 802.1X: A specification for Layer-2 authentication and access control. A host which attaches to a local network using 802.1X must provide authentication credentials before it can gain Layer-2 access to that local network.

IEEE EUI-64 format: A format for globally unique 64-bit numbers associated with a network interface. EUI stands for “Extended Unique Identifier.” EUI-64 format is used by IPv6 for interface identifiers generated by an underlying IEEE 802 MAC address. These interface identifiers are generated by inserting 16 well-known bits in between the first and last 24 bits of the 48-bit IEEE MAC address. Also see hardware address.

ingress filtering: An admission control method in which traffic entering a network is filtered out if it contains an illegal or unexpected source address. Ingress filtering is employed on edge routers to prevent traffic with spoofed addresses from entering the network. Also see RPF.

interface identifier: In the IPv6 specification, the last 64 bits of the 128-bit global unicast address. The interface identifier uniquely addresses hosts in a Layer-2 network segment. The interface identifier may be assigned manually, but may be derived from the interface’s underlying hardware address for purposes of auto-configuration. Also see hardware address, IEEE EUI 64 format.

intermediate system: A network device which is situated in the interior of a network. Intermediate systems forward network traffic as opposed to end systems, which are the endpoints for most network traffic. Routers, firewalls, proxies, and gateways are all intermediate systems.

IPsec: An Internet standard for securing the IP layer (Layer-3) of network communications. Security services offered by IPsec include authentication, integrity, and confidentiality of IP traffic. IPsec has two modes: tunnel and transport mode. Transport mode, in which an end system may secure IP traffic between itself and another end system, offers the possibility of “end-to-end” security. IPsec is mandatory in IPv6 but optional in IPv4.

link-local address: In the IPv6 specification, a unicast address whose scope is local. Traffic to and from link-local addresses may be used for communication between local hosts, but it may not exit the local network segment. Link-local addresses are autonomously and automatically generated by network interfaces and are critical for local network functions, including address resolution, next-hop resolution, and router discovery (see separate entries). Link-local addresses are generated from a well-known prefix and the interface identifier.

logical address: Here, the Layer-3 network address (the IP address). The term “logical address” is used to distinguish the IP address from the Layer-2 hardware address.

multicast addresses: IP addresses which specify a set of destinations (a “multicast group”). Multicast addresses in IPv6 use a well-known prefix. There are a few special multicast addresses which are used for local network functions. These include the “all-hosts” and “all-routers” multicast groups, as well as the “solicited-node” multicast address, which is used in the Neighbor Solicitation (NS) message for address resolution. The solicited-node multicast address is generated from a well-known prefix plus the last 24 bits of the solicited node’s unicast address. Also see address resolution.

NAT: network address translation (RFC1631). An IPv4 specification for translating IP addresses in a network device. The purpose of NAT was originally to conserve scarce public IP address resources, since it allows a network inside of the translating device to map internal private IP addresses to globally reachable public IP addresses. A NAT device modifies the address fields in packet headers (translation) and maintains a translation state table in order to keep track of end-to-end connections (for TCP) or pseudo-connections (for UDP). A variant of NAT (Network Address Port Translation, or NAPT) allows a single outward facing public IP address to represent multiple internal private IP addresses. NAT has been criticized for its role in the reduction of end-to-end transparency on the Internet. One major effect of this problem is the incompatibility of NAT and IPsec. Workaround solutions such as “NAT Traversal (NAT-T)” have been designed to get around this incompatibility.

ND: Neighbor Discovery protocol. A formal set of functions which perform various operations among hosts and routers on a local link. Key functions include router discovery, address resolution, next-hop determination, neighbor unreachability detection, and duplicate address detection.

next-hop determination: A local network function in which a node determines whether or not the destination IP address in a datagram is local and should be forwarded locally or is non-local and should be forwarded to a router. In IPv4, this functionality is carried out by the application of subnet masks to the IP address. In IPv6, next-hop determination is accomplished by comparing the local prefix information (received from a router in the RA message) with the destination address in the datagram.

OSI reference model: Open System Interconnection model. An abstraction used to describe layered network protocol design. A layer specified in the OSI model describes the set of functions exclusively performed by that layer. Each layer provides services to higher layers and requests services of lower layers. In IP networks, the most commonly referenced layers are “Layer-2,” which is responsible for local node-to-node traffic delivery (e.g., IEEE 802 protocols) and “Layer-3, which is responsible for end-to-end traffic delivery (e.g., IP protocols).

peering: The interconnection of administratively separate networks for the purpose of exchanging traffic. Peering is voluntary and involves no payments. Peering on the Internet is implemented through the use of the BGP external routing protocol. The core of the Internet consists of Tier-1 ISPs which have multiple peering relationships with each other.

PKI: public key infrastructure. A system in which a trusted third-party manages identities of users, systems, and other entities through various cryptographic mechanisms, including public key cryptography, digital signatures, and certificates. The trusted third party in the PKI will establish and attest to the veracity of identities. The attestations are normally made in the form of digitally signed certificates, which bind public keys to entities.

packet header: In IP, the packet header is at the beginning of a packet and contains data used for forwarding the packet. The header is distinguished from the payload portion of the packet, which contains the data being forwarded. The IPv4 header varies in length depending on the use of various options. In contrast, the IPv6 header is of fixed length and uses optional “extension headers” in order to implement options. IP headers contain source and destination addresses, among other data fields.

perimeter model: A model of security which emphasizes maintaining strict boundaries between the “outside” and “inside” of a network. The perimeter model is based on the assumption that network traffic entering and exiting a network can be controlled and monitored at a few key chokepoints (e.g., firewalls, application proxies, and/or NAT devices). A common criticism of the perimeter model is that it results in lax security for systems in the interior of a network.

policy object: In a network device, an abstraction which allows descriptors such as IP addresses and protocol ports to be grouped together for purposes of simplifying configured policy. For instance, a policy object may consist of a list of otherwise unrelated IP addresses that are permitted access to a certain resource. Changes to policy would involve modifying the configurations applying to the object and not the individual IP addresses making up that object.

prefix: The portion of an IP address used for forwarding purposes. In IPv4, the prefix is derived by applying a subnet mask to the IP address to determine the number of prefix bits. Longer prefixes specify smaller and more specific networks. In IPv6, the global routing prefix is the first 48 bits of the address.

protocol stack: The software on a system which implements network protocols. Protocol stacks implement the layered network functions defined in the OSI reference model, although they may not adhere strictly to the specified layers. Also see OSI reference model.

QoS: quality of service. In IP networks, QoS refers to the set of mechanisms which attempt to provide some guarantee of service level to specified traffic flows. Service level may be defined by a number of network traffic characteristics, such as bit rate, latency, jitter, and error rate. QoS mechanisms include signaling, queuing, policing, and congestion avoidance methods.

router discovery: A local network function which allows hosts to automatically discover routers and router information on the local network segment. In IPv4, router discovery is implemented in the ICMP Router Discovery Protocol (IRDP); in IPv6, router discovery is implemented by the Router Solicitation (RS) and Router Advertisement (RA) messages in the Neighbor Discovery (ND) protocol.

routing table: A table on a system which specifies next-hop gateways for network destinations. The routing table consists of entries that specify a network prefix and a corresponding next-hop gateway for that prefix (as well as a few other fields). End systems usually have small routing tables which specify a default gateway. Routers have large tables which are updated in real time by routing protocols. Core ISP routers contain the global routing table, which contains hundreds of thousands of IPv4 prefixes.

RPF (unicast): Reverse-path forwarding (unicast) or Reverse-path filtering. An admission control method used to prevent address spoofing. RPF implements ingress filtering by ensuring that the source addresses of packets entering an interface correspond to legitimate routes out of that interface (the “reverse path”). Also see ingress filtering.

scope: An attribute of an IP addressing type which specifies the range of use for that type. Global and local are two scopes defined in IPv6. Globally-scoped IP addresses are intended to be reachable throughout the global Internet. Locally-scoped addresses should only be reachable from the local link. Link-local addresses are inherently local in scope.

SNMP: Simple Network Management Protocol. The standard Internet method for communicating management information about network nodes. SNMP employs both managing software and agent software resident on nodes to be managed. SNMP specifies messaging format for communicating the management information as well as a framework for representing the management information (the management information base, or MIB).

solicited-node multicast address: See multicast addresses.

spoofed addresses: A common method used to mask the source of attacks involving IP traffic. There is currently no practical globally deployable method to authenticate the source address in an IP packet. As a result, attackers can spoof the source address field of a packet for use in various attacks. Various authentication mechanisms will be required to bind a source address to a legitimate principal. Also see ingress filtering.

subnet identifier: In the IPv6 specification, the 16 bits of the 128-bit global unicast address following the global routing prefix and preceding the interface identifier. The subnet identifier allows the end-user organization to address over 64,000 network segments for internal use.

subnet mask: In IPv4, a 32-bit structure used to interpret 32-bit IP addresses. A subnet mask consists of a variable series of 1s followed by a series of 0s. The subnet mask is applied to the

IP address through a “bitwise AND” operation to extract the network prefix associated with that IP address.

Tier-1 ISP: An internet service provider which operates at the top layer of the global routing hierarchy. Tier 1 ISPs peer with other Tier 1 ISPs and provide service for downstream ISPs and end-user organizations.

tunneling interface: A virtual interface on a system used for tunneling network traffic. An IP packet sent to a tunneling interface is encapsulated (“tunneled”) within another IP packet for delivery to another tunneling endpoint, where the packet will be decapsulated for normal delivery. Tunneling interfaces are one solution for the co-existence of IPv4 and IPv6. In a tunneling solution, “clouds” of IPv6 networks would be tunneled through an existing IPv4 infrastructure. Also see dual stack.

References

URLs are valid as of the publication date of this document.

1 Autonomous Agents: The Good, the Bad, and the Underutilized

[Barber 03] Barber, K. Suzanne & Park, Jisun. “Autonomy Affected by Beliefs: Building Information Sharing Networks with Trustworthy Providers.” *Second International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2003).* Melbourne, Australia, Jul. 14-18, 2003. Austin. TX: University of Texas, 2003. http://www.lips.utexas.edu/~jisun/academic/academic_files /TR_ACD_AAMAS04.pdf.

[Bernard 99] Bernard, Douglas, et al. “Spacecraft Autonomy Flight Experience: The DS1 Remote Agent Experiment,” (AIAA-99-4512). *AIAA Space Technology Conference & Exposition.* Albuquerque, NM: Sep. 28-30, 1999. Menlo Park, CA: American Institute of Aeronautics and Astronautics, 1999. http://ic.arc.nasa.gov/projects/remote-agent/AIAA-99.pdf.

[Bradshaw 97] Bradshaw, Jeffrey M. *Software Agents*. Menlo Park, CA: AAAI Press, 1997 (ISBN 0-262-52234-9).

[Brustoloni 91] Brustoloni, Jose C. *Autonomous Agents: Characterizations and Requirements* (CMU-CS-91-204). Pittsburgh, PA: School of Computer Science, Carnegie Mellon University, 1991.

[Carrol 02] Carroll, Charles M. “Population and Maintenance of an Intelligence Database Utilizing Intelligent Agent Technologies.” (MS thesis). Monterey CA: Naval Postgraduate School, 2002. http://www.cs.nps.navy.mil/people/faculty/rowe /carrollthesis/carrollthesisunclass.htm.

[Dastani 03] Dastani, Mehdi; Dignum, Frank; & Meyer, John-Jules. “Autonomy and Agent Deliberation.” *First International Workshop on Computational Autonomy - Potential, Risks,* *Solutions*. Melbourne, Australia, Jul. 14, 2003. *Lecture Notes in Computer Science 2969*, 0302-9743 (2004): 114-127. http://springerlink.metapress.com/content/mgdbcad9ech0y3ny/fulltext.pdf.

[Franklin 03] Franklin, Stan. “An Autonomous Software Agent for Navy Personnel Work: A Case Study,” 60-65. *AAAI Spring Symposium*. Stanford, CA, Mar. 24-26, 2003. Menlo Park, CA: American Institute of Aeronautics and Astronautics Press, 2003.

[Hoffman 98] Hoffman, Martin O.; McGovern, Amy; & Whitebread, Kenneth R. “Mobile Agents on the Digital Battlefield,” 219-225. *Proceedings of the Second International Conference on Autonomous Agents*. Minneapolis/Saint Paul, MN, May 10-13, 1998. New York, NY: ACM, 1998. http://doi.acm.org/10.1145/280765.280805.

[Kephart 03] Kephart, Jeffery O. & Chess, David M. “The Vision of Autonomic Computing.” *IEEE Computer 36*, 1 (Jan. 2003): 41-50. http://ieeexplore.ieee.org/iel5/2/26009 /01160055.pdf.

[Kubrick 68] Kubrick, Stanley. *2001: A Space Odyssey*. Borehamwood, England: Metro-Goldwyn-Mayer British Studios, 1968.

[Maes 94] Maes, Pattie. “Modeling Adaptive Autonomous Agents.” *Artificial Life 1*, 1-2 (Fall 1993/Winter 1994): 135-162.

[Perme 03] Perme, David; Whelan, Mark; & Loftus, William P. “Achieving Interoperability of Command and Control Systems Using Translation Gateways.” *Information and Security, 10*, 1 (2003): 97-104. http://www.isn.ch/pubs/ph/details.cfm?id=543.

[Qiu 05] Qiu, Xuemei. “Agent Interaction in A Semantic Web Environment: A State-Of-The-Art Survey And Prospects In Knowledge Mobilization.” *Information Systems in Scandinavia (IRIS’28).* Kristiansand, Norway Aug. 6-9, 2005. http://www.hia.no/iris28/Docs/IRIS2028-1088.pdf. Quoting H.S. Nwana & D.T. Ndumu. “A Perspective on Software Agents Research.” *Knowledge Engineering Review 14*, 2, (Sep. 1999): 125-142. Available through http://journals.cambridge.org/action/displayJournal?jid=KER.

[Rudowsky 05]	Rudowsky, I.; Kulyba, O.; Kunin, M.; Ogarodnikov, D.; Raphan, T. Title: “Managing a Relational Database with Intelligent Agents,” 238-242. *ITCC 2005: International Conference on Information Technology: Coding and Computing* Las Vegas, NV, Apr. 4-6, 2005. New York, NY: IEEE Digital Library, 2006.
[Russell 95]	Russell, Stuart & Norvig, Peter. *Artificial Intelligence: A Modern Approach*. Englewood Cliffs, NJ: Prentice-Hall, 1995 (ISBN 0-131-03805-2).
[Turing 50]	Turing, Alan. “Computing Machinery and Intelligence.” *Mind LIX*, 236 (Oct. 1950): 433-460. http://mind.oxfordjournals.org/cgi/reprint/LIX/236/433.
[Wooldridge 95]	Wooldridge, Michael & Jennings, Nicolas. “Intelligent Agents: Theory and Practice.” *Knowledge Engineering Review 10,* 2 (1995): 115–152.

2 Component Testing and Its Limitations When Applied to a System of Systems

[Alexander 99]	Alexander, Roger T. & Blackburn, Mark R. *Component Assessment Using Specification-Based Analysis and Testing.* Software Productivity Consortium, May 1999. Available through request to Ask-SSCI@systemsandsoftware.org.
[DAU 04]	Defense Acquisition University. *Defense Acquisition Guidebook*. 2004. http://akss.dau.mil/dag.
[DISA 97]	Defense Information Systems Agency. *DITSCAP Primer*. 1997. http://www.certconf.org/presentations/2005/files/WD3.pdf.
[DoD 97]	Department of Defense Information Technology Security *Certification and Accreditation Process*, 1997. http://iase.disa.mil/ditscap/DitscapFrame.html.
[Haines 97]	Haines, Gary, Carney, David, Foreman, John. “Component-Based Software Development/COTS Integration.” SEI Technology Description. Pittsburgh PA: Software Engineering Institute, Carnegie Mellon University, 1997. http://www.sei.cmu.edu/str/descriptions/cbsd_body.html.

[IEEE 98] Institute of Electrical and Electronic Engineers. ISO/IEC 14598 Information Technology – Software Product Evaluation – Part 5: Process for Evaluators. 1998. Available through http://www.techstreet.com/cgi-bin /detail?product_id=862743.

[Lund 03] Lund, Mass Soldal. *Testing Decomposition of Component Specifications Based on a Rule for Formal Verification.* New York, NY: IEEE Press, 2003. http://heim.ifi.uio.no/~ketils/sardas/031116.qsic03.pdf.

[McGregor 97] McGregor, John D. "Component Testing." *Journal of Object Oriented Programming*, Mar.-Apr. 1997. http://www.cs.clemson.edu /~johnmc/joop/col3/column3.html.

[Maier 96] Maier, M. "Architecting Principles for Systems of Systems." *Proceedings of the Sixth Annual International Symposium, International Council on Systems Engineering*, Boston, MA, 1996: 567- 574, www.infoed.com/Open/PAPERS/systems.htm.

[Michael 05] Michael, C. & Radosevich, W. "Risk-based and Functional Security Testing." https://buildsecurityin.us-cert.gov /daisy/bsi/articles/best-practices/testing/255.html.

[NSA CSS 06] National Security Agency Central Security Service. "Global Information Grid." http://www.nsa.gov/ia/industry/gig.cfm?MenuID=10.3.2.2 (2006).

[Oberndorf 97] Oberndorf, Tricia. "COTS and Open Systems – An Overview" SEI Technology Description, 1997. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University. http://www.sei.cmu.edu/str/descriptions/cots_body.html.

[SEI 05] Software Engineering Institute "Component-Based Software Development/COTS Integration." SEI Technology Description Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University. http://www.sei.cmu.edu /str/descriptions /cbsd_body.html (2005).

[Valetto 95] Valetto, G. & Kaiser, G.E. "Enveloping Sophisticated Tools into Computer-Aided Software Engineering Environments," 40-48. *Proceedings of 7th IEEE International Workshop on CASE.* Toronto, Ontario, Canada, Jul. 10-14, 1995. Los Alamitos, CA: IEEE Computer Society Press, 1995.

[Wallnau 04] Wallnau, Kurt C. *Software Component Certification: 10 Useful Distinctions* (SEI/CMU-2004-TN-031, ADA430991). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2004. http://www.sei.cmu.edu/publications /documents/04.reports/04tn031.html.

[Webster 96] Webster, N. *Webster's New Universal Unabridged Dictionary.* New York, NY: Barnes and Noble Books, 1996.

[Wikipedia 06] Wikipedia. "Global Information Grid." 2006. http://en.wikipedia.org/wiki/Global_Information_Grid.

3 Evaluating Emergent Issues Resulting from Unanticipated Consequences

[Boutelle 96] Boutelle, S. & Filak, R. "AFATDS: The Fire Support Window to the 21st Century." *Joint Force Quarterly, 11* (Spring 1996): 16-21. http://www.dtic.mil/doctrine/jel/jfq_pubs /jq019604.pdf.

[FAS 96] Federation of American Scientists (FAS). *Theater Battle Management Core System Concept of Operations*. http://www.fas.org /man/dod-101/usaf/docs/tbmcs/CON-OPS.html (1996).

[FAS 98] Federation of American Scientists (FAS). *Advanced Field Artillery Tactical Data System (AFATDS).* http://www.fas.org /man/dod-101 /sys/land/afatds.htm (1998).

[Fisher 01] Fisher, David A. & Lipson, Howard F. "Emergent Algorithms: A New Method for Enhancing Survivability in Unbound System." *Proceedings of the 32nd Annual Hawaii International Conference on System Sciences*. Maui, HI, Jan. 5-8, 1999. Los Alamitos, CA: IEEE Computer Society Press, 1999. http://ieeexplore.ieee.org/xpls/abs_all.jsp?isnumber =16787&arnumber=772824&count=68&index=42.

[Garlan 95] Garlan, D.; Allen, R.; & Ockerbloom, J. "Architectural Mismatch: Why Reuse is So Hard." *IEEE Software 12*, 6 (Nov. 1995): 17-26. http://ieeexplore.ieee.org/iel1/52 /9910/00469757.pdf?tp=&arnumber=469757&isnumber=9910.

[GlobalSecurity.org 05] GlobalSecurity.org. “Advanced Field Artillery Tactical Data System (AFATDS).” http://www.globalsecurity.org/military /systems/ground/afatds.htm (2005).

[GlobalSecurity.org 05] GlobalSecurity.org. *Theater Battle Management Core Systems*. http://www.globalsecurity.org/military/systems /aircraft/systems/tbmcs.htm (2005).

[Hines 05] Hines, P.; Liao, H.; Jia, D.; & Talukdar, S. “Autonomous Agents and Cooperation for the Control of Cascading Failures in Electric Grids,” 273–278. *Proceedings of the IEEE Conference on Networking, Sensing, and Control*. Tucson, AZ, Mar. 19-22, 2005. Los Alamitos, CA: IEEE Computer Society Press, 2005. Available through http://ieeexplore.ieee.org/xpls/abs_all.jsp?isnumber= 31421&arnumber=1461200&count =194&index=48.

[Lockheed Martin 05] Lockheed Martin Corporation. “Lockheed Martin’s TBMCS Command and Control System for Air Combat Approved for U.S. Forces.” 2005. http://www.lockheedmartin.com/wms /findPage.do?dsp= fec&ci=12302&rsbci=0&fti=0&ti=0&sc=400.

[Miller 01] Miller, A.; Jefferson, M.; & Rodgers, J. “Global Information Grid Architecture.” *The Edge 5*, 2 (Jul. 2001): 8-9. http://www.mitre.org/news/the_edge/july_01/miller.html.

[NSA 05] National Security Agency. “The GIG Vision Enabled by Information Assurance.” http://www.nsa.gov/ia/industry /gig.cfm?%20MenuID%20 =10.3.2.2 (2005).

[Raytheon 05] Raytheon Company. “Advanced Field Artillery Tactical Data System (AFATDS),” 2005. http://www.raytheon.com /products/afatds.

[Schneider 99] Schneider, F.; Bellovin, S.; & Inouye, A. “Building Trustworthy Systems: Lessons from the PTN and Internet.” *IEEE Internet Computing 3*, 6 (Nov. 1999): 64-72. Available through http://portal.acm.org/citation.cfm?id=613461&dl=&coll =GUIDE&CFID=15151515&CFTOKEN=6184618.

[Uchitel 00] Uchitel, S. & Yankelevich, D. “Enhancing Architectural Mismatch Detection with Assumptions,” 138-146. *Proceedings, Seventh Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS'00*). Edinburgh, UK, Apr. 4-7, 2000. Los Alamitos, CA: IEEE Computer Society Press, 2000. Available through http://ieeexplore.ieee.org/xpls/abs_all.jsp?isnumber =18055&arnumber=839871&count=51&index=16.

4 IPv6 Network Addressing: Architectural Properties and Operational Survivability Effects

[Abley 06] Abley, J. & Lindqvist, K. *Operation of Anycast Services.* IETF Internet Draft. The Internet Society, Jan. 2006. http://www.ietf.org/internet-drafts/draft-ietf-grow-anycast-04.txt.

[ARIN 02] American Registry for Internet Numbers (ARIN). *Ipv6 Address Allocation and Assignment Policy.* http://www.arin.net/policy/archive/ipv6_policy.html (2002).

[Arkko 05] Arkko, E.; Kempf, J.; Zill, B.; & Nikander, P. *Secure Neighbor Discovery (SEND).* IETF Request for Comments 3971. The Internet Society, Mar. 2005. http://tools.ietf.org/wg/mipshop /draft-haddad-mipshop-optisend-02.txt.

[Bound 06] Bound, J.; Pouffary Y.; Klynsma, S.; Chown, T.; & Green, D. *IPv6 Enterprise Network Analysis*. IETF Internet Draft. The Internet Society, Feb. 2006. http://www.ietf.org/internet-drafts/draft-ietf-v6ops-ent-analysis-06.txt.

[Carpenter 00] Carpenter, B. *Internet Transparency*. IETF Request for Comments 2775. The Internet Society, Feb. 2000. http://www.ietf.org/rfc/rfc3093.txt.

[Chown 05] Chown, T. *IPv6 Implications for TCP/UDP Port Scanning.* IETF Internet Draft. The Internet Society, Oct. 2005. http://www.6net.org/publications/standards/draft-chown-v6ops-port-scanning-implications-00.txt.

[Chown 06] Chown, T.; Ford, A.; & Venaas, S. *Things to Think about when Renumbering an IPv6 Network.* IETF Internet Draft. The Internet Society, Mar.2006. http://www.ietf.org/internet-drafts/draft-chown-v6ops-renumber-thinkabout-05.txt.

[Comer 00] Comer, D. *Internetworking with TCP/IP Vol.1: Principles, Protocols, and Architecture, 4th ed*. Upper Saddle River, NJ: Prentice Hall, 2000.

[Conta 06] Conta, A.; Deering, S.; & Gupta, M. *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification.* IETF Request for Comments 4443. The Internet Society, Mar. 2006. http://www.rfc-editor.org/rfc/rfc2463.txt.

[DARPA 01] Defense Advanced Research Projects Agency. *Considerations in Developing Survivable Architectures for Global Information Grid (GIG) Systems.* Aug. 2001. http://www.ai.mit.edu/projects /tuesday-group/SGIGReport-Final.pdf.

[Davies 06a] Davies, E. & Mohacsi, J. *Best Current Practice for Filtering ICMPv6 Messages in Firewalls.* IETF Internet Draft. The Internet Society, Mar. 2006. http://www3.ietf.org/proceedings/06mar/IDs /draft-ietf-v6ops-icmpv6-filtering-bcp-01.txt.

[Davies 06b] Davies, E.; Krishnan, S.; & Savola, P. *IPv6 Transition/Co-existence Security Considerations. IETF Internet Draft*. The Internet Society, Mar. 2006. http://www.6net.org/publications /standards/draft-savola-v6ops-security-overview-03.txt.

[Dondeti 01] Dondeti, L.; Hardjono, T.; & Haberman, B. *Security Requirements of IPv6 Anycast.* IETF Internet Draft. The Internet Society, Jun. 2001. http://www.join.uni-muenster.de/Dokumente/drafts/draft-dondeti-ipv6-anycast-security-00.txt.

[Draves 03] Draves, R. *Default Address Selection for Internet Protocol version 6 (IPv6).* IETF Request for Comments 3484. The Internet Society, Feb. 2003. http://www.ietf.org/rfc/rfc3484.txt.

[Egevang 94] Egevang, K. & Francis, P. *The IP Network Address Translator (NAT).* IETF Request for Comments 1631. The Internet Society, May 1994. http://www.ietf.org/rfc/rfc1631.txt.

[Fuller 93] Fuller, V.; Li, T.; Yu, J.; & Varadhan, K. *Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy.* IETF Request for Comments 1519. The Internet Society, Sep. 1993. http://www.rfc-archive.org/getrfc.php?rfc=4632.

[Hagino 01] Hagino, J. & Ettikan, K. *An Analysis of IPv6 Anycast*. IETF Internet Draft. The Internet Society, Feb. 2001. http://www.ietf.org/proceedings/01mar/I-D/1id-abstracts.txt.

[Hain 00] Hain, T. *Architectural Implications of NAT*. IETF Request for Comments 2993. The Internet Society, Nov. 2000. http://www.ietf.org/rfc/rfc2993.txt.

[Hinden 98] Hinden, R. & Deering, S. *IPv6 Multicast Address Assignments*. IETF Request for Comments 2375. The Internet Society, Jul. 1998. http://www.ietf.org/rfc/rfc2375.txt.

[Hinden 03] Hinden, R.; Deering, S.; & Nordmark, E. *IPv6 Global Unicast Address Format*. IETF Request for Comments 3587. The Internet Society, Aug. 2003. http://rfc.net/rfc3587.html.

[Hinden 05] Hinden, R. & Haberman, B. *Unique Local IPv6 Unicast Addresses*. IETF Request for Comments 4193. The Internet Society, Oct. 2005. http://www.ietf.org/rfc/rfc4193.txt.

[Hinden 06] Hinden, R. & Deering, S. *IP Version 6 Addressing Architecture*. IETF Request for Comments 4291. The Internet Society, Feb. 2006. http://www.rfc-editor.org/rfc/rfc3513.txt.

[Huston 05] Huston, G. *Architectural Approaches to Multi-homing for IPv6*. IETF Request for Comments 4177. The Internet Society, Sep. 2005. http://rfc.sunsite.dk/rfc/rfc4177.html.

[IEFT 06] Internet Engineering Task Force. *V6ops Status Pages* http://tools.ietf.org/wg/v6ops/ (2006).

[JP 3-09 03] Joint Chiefs of Staff. *Joint Tactics, Techniques, and Procedures for Close Air Support (CAS).* Joint Doctrine Reference Publication 3-09. Sep. 2003. Washington, DC: Defense Technical Information Center, 2003. http://stinet.dtic.mil/oai/oai?&verb=getRecord&metadataPrefix=html&identifier=ADA429336.

[Knight 00] Knight, J.; Sullivan, K.; Elder, M.; & Wang, C. “Survivability Architectures: Issues and Approaches.” *Proceedings of the DARPA Information Survivability Conference and Exposition*. Los Alamitos, CA, Jan. 2000. New York, NY: IEEE Computer Society Press, 2000. http://www.cs.virginia.edu/~jck/publications/discex2000.pdf.

[Knight 03] Knight, J.; Strunk, E.; & Sullivan, K. “Towards a Rigorous Definition of Information System Survivability.” *Proceedings of the DARPA Information Survivability Conference and Exposition*. Washington DC, Apr. 2003. New York, NY: IEEE Computer Society Press, 2003. http://dependability.cs.virginia.edu/publications /discex.2003.pdf.

[Kozierok 05] Kozierok, C. *TCP/IP GUIDE Version 3.0*. Sep. 2005. Available through http://www.tcpipguide.com.

[Lipson 02] Lipson, H. *Tracking and Tracing Cyber-Attacks: Technical Challenges and Global Policy Issues* (CMU/SEI-2002-SR-009). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon

University, 2002. http://www.sei.cmu.edu/publications /documents/02.reports/02sr009.html.

[Mead 00] Mead, N.; Ellison, R.; Linger, R.; Longstaff, T.; & McHugh, J. *Survivable Network Analysis Method* (CMU/SEI-2000-TR-013, ADA383771). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000. http://www.sei.cmu.edu /publications/documents/00.reports /00tr013.html.

[Narten 98] Narten, T.; Nordmark, E.; & Simpson, W. *Neighbor Discovery for IP Version 6 (IPv6).* The Internet Society, 1998. http://www.ietf.org/rfc/rfc2461.txt.

[Narten 01] Narten, T. & Draves, R. *Privacy Extensions for Stateless Address Autoconfiguration in IPv6*. IETF Request for Comments 3041. The Internet Society, 2001. http://www.ietf.org/rfc/rfc3041.txt.

[Nikander 04] Nikander, P.; Kempf, J.; & Nordmark, E. IPv6 *Neighbor Discovery (ND) Trust Models and Threats.* IETF Request for Comments 3576. The Internet Society, 2004. http://rfc.sunsite.dk/rfc/rfc3756.html.

[Roese 04] Roese, J. *Internet Protocol Version 6.* White Paper. Enterasys, 2004. http://www.enterasys.com/products/whitepapers/ipv6_wp.pdf.

[Sterbenz 02] Sterbenz, J.; Krishnan, R.; Hain, R.; Jackson, A.; Levin, D.; Ramanathan, R.; & Zao, J. “Survivable Mobile Wireless Networks: Issues, Challenges, and Research Directions.” *Proceedings of the Third ACM Workshop on Wireless Security.* Atlanta, GA, Sep. 2002. New York, NY: ACM Press.

[Thomson 98] Thomson, S. & Narten, T. *IPv6 Stateless Address Autoconfiguration*. IETF Request for Comments 2462. The Internet Society, Dec. 1998. http://www.ietf.org/rfc/rfc2462.txt.

[USAF 01] U.S. Air Force Command and Control. *Air Force Doctrine Document 2-8*. Feb. 2001. http://www.dtic.mil/doctrine/jel/service_pubs/afd2_8.pdf.

[Van de Velde 05] Van de Velde, G.; Hain, T.; Droms, R.; Carpenter, B.; & Klein, E. *IPv6 Network Architecture Protection.* The Internet Society, Oct. 2005. http://www.6net.org/publications /standards/draft-vandevelde-v6ops-nap-01.txt.

[Weber 04] Weber, S. & Cheng, L. "A Survey of Anycast in IPv6 Networks." *IEEE Communications Magazine,* Jan. 2004.

[Zhang 01] Zhang, Y.; Dao, S.; Vin, H.; Alvisi, L.; & Lee, W. "Heterogeneous Networking: A New Survivability Paradigm." *ACM New Security Paradigms Workshop Conference Report*, Cloudcroft, NM: Sep.10-13, 2001. New York, NY: ACM Press, 2001.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY
(Leave Blank)

2. REPORT DATE
March 2007

3. REPORT TYPE AND DATES COVERED
Final

4. TITLE AND SUBTITLE
Global Information Grid Survivability: Four Studies

5. FUNDING NUMBERS
FA8721-05-C-0003

6. AUTHOR(S)
Richard D Ciampa, Dawn Day, Jennifer R. Franks, Christopher T. Tsuboi

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

8. PERFORMING ORGANIZATION
REPORT NUMBER
CMU/SEI-2006-SR-008

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
HQ ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2116

10. SPONSORING/MONITORING AGENCY
REPORT NUMBER

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified/Unlimited, DTIC, NTIS

12B DISTRIBUTION CODE

13. ABSTRACT (MAXIMUM 200 WORDS)

The four studies in this document are student contributions to the SEI Global Information Grid (GIG) Survivability Study. Each study explores an issue relevant to the survivability of networks which are systems of systems. Since the GIG is inherently a system of systems, the survivability of operational concepts such as Joint Battle Management Command and Control (JBMC2) will largely depend on the extent to which GIG architecture is approached from this perspective. Systems of systems differ from large, monolithic systems because of the simultaneous independence and interdependence of their constituent parts, and therefore traditional survivability methods are not sufficient. To deal with the operational complexity resulting from qualities peculiar to systems of systems, planners and builders of the GIG will need to formulate broad strategic approaches taking these qualities into account. These four studies have attempted to identify characteristics of systems of systems which may be useful in this endeavor. The specific areas explored in this document include the following: the applicability of autonomous agents in a system of systems; the suitability of conventional software testing in a system-of-systems environment; emergent properties and unanticipated consequences in a system of systems; the role of ontologies in systems-of-systems interoperability; the architectural properties and operational survivability effects of internet protocol version 6 (IPv6) technology.

14. SUBJECT TERMS
autonomous agents, critical success factors, component testing, component certification, DTSCAP, emergent issues, emergent behaviors, GIG system integration, IPv6 network addressing, architectural properties, operational survivability, systems of systems

15. NUMBER OF PAGES
96

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT
Unclassified

18. SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

19. SECURITY CLASSIFICATION OF ABSTRACT
Unclassified

20. LIMITATION OF ABSTRACT
UL